Smart Mirror

An interactive touch-free mirror that maximizes time efficiency and productivity

Sponsored by Central Florida Inpatient Medicine

Group 10
Senior Design II - Project Documentation

4-28-2014

Georgiy Brussenskiy Christopher Chiarella Vishal Nagda

Table of Contents

1.0 Executive Summary

Efficiency and productivity are two qualities that are increasingly establishing their dominance as keywords companies are using to market their products. The fact that their product can multitask or increase productivity better than the competition has become a real selling point. This is due to the fact that effective time management is an essential factor in increasing production of day-to-day life. The best time management strategies involve being able to find time where there was no time before. Integration of technology into people's daily lives has made that time management possible. The use of products such as tablets, PCs, and smartphones have given people access to the tools needed to be productive.

However, though successful technological products have been used to increase productivity, it has done its fair share to stifle it as well. The use of technology has become another task on everyone's daily to-do list. Technology should mold to our schedule, not the other way around. That is where the smart mirror idea originated. The smart mirror idea aimed to integrate technology seamlessly into people's lives by putting it where everyone's routine eventually collides, the bathroom. The goal of the smart mirror is to increase a user's productivity by saving them time. The smart mirror provides a near effortless experience that allows the user to just walk up and be greeted with information they would typically need another device for. Despite the fact this information can be found on the user's other devices, it's the time-saving convenience of having this information available during the typical bathroom routines.

The smart mirror has the necessary applications and features needed for time efficiency focused device. First, there is the easy on and off. The mirror automatically recognizes that there is a user present and turns on the screen hidden behind the two-way mirror. A suit of six applications are presented on the 32" high-definition screen surrounded by a framed encasing that contains speakers and a webcam. These apps include weather, Twitter, news, to-do list, calendar, and music. The smart mirror application is run from a central computer with Windows 8.1 which features a multi-core CPU, solid-state memory, and wireless connectivity. The application can be interacted with by use of the Leap Motion controller. This controller allows for touch-free control of the smart mirror by use of finger swipes, taps, and circles. There is also the option to use voice control to play music, post a tweet, or add a task to the to-do list. Finally, the mirror monitors its own temperature and humidity levels so that if the bathroom becomes potentially harmful to the computer's hardware, it can execute preventive measures and shut the system down.

To develop a design of this size, significant research had to be done. First, similar projects and products had to be investigated to what has been done and possibly what needs to be done. Next, the individual pieces of the mirror had to research in order to decide on the best product to use in the smart mirror system. Some of these topics included the gesture control, voice recognition system, MCU, and hard drive type. Once the comparisons were made, each component was analyzed on how it will function within the system and what it has to offer.

The design of the smart mirror was broken down into subsystems that represent important components of the project. Since there are hardware and software components to design, a block diagram was needed for both. The technical specifications were written for each subsystem for documentation and reference during the prototyping stage. The design contained charts, state-diagrams, and circuit designs regarding the details relating to what was prototyped. Once completed, the smart mirror delivered the experience of technology becoming part of a user's routine and not an extra task.

2.0 Project Description

2.1 Motivation

Effective time management is one of the most important factors for success and productivity in a person's day-to-day life. With the increasing integration of technology in our lives, maintaining an efficient schedule has become both easier and more difficult. Keeping up to date with appointments, Twitter, news, social media, and other things is made easier through technology such as tablets, PCs, and smartphones yet also provide distractions that can interrupt anyone's routine. Technology has become another task in the day that time must be allotted for. In the finite time of the day, technology needs to be designed to work within our schedule and not be an extra piece to it.

The key to effective time management involving technology is multitasking. Anyone in the business or academic world would agree that every second counts in the day. This project was formulated through inspiration seen through movies such as Iron Man and tech demos, such as Samsung's transparent LCD Smart Window, seen at the International Consumer Electronics Show in 2012. This extends as well to the continuing trend of integrating touchscreens and internet-connectivity into everyday appliances such as ovens and refrigerators. The idea of a smart home is the direction lots of companies are heading and while the kitchen has been getting lots of attention, the bathroom has not. Besides the kitchen, the bathroom is one of the busiest rooms in the home, so it is an excellent place to expand the smart home next.

Constant information and instant access to it drive the current generation. Forget bringing smartphones and tablets into the bathroom and risking damage. The smart mirror will show you that information with the swipe of a hand. The smart mirror is the result of our team brainstorming on how to solve all these issues and develop something that is functional as well as a showpiece.

2.2 Goals and Objectives

The smart mirror must offer benefits of using modern technology while integrating seamlessly into the standard bathroom routines of most people. The smart mirror must be simple and as intuitive as possible. The smart mirror would be used to merge technology and the need for information into anyone's daily schedule. With the mirror in place, the

user could interact and obtain the information they want during their normal morning and night bathroom routines.

This smart mirror aims to reduce and possibly eliminate the need for the user to make time in their daily morning or nightly routine to check their PC, tablet, or smartphone for the information they need. The mirror will provide the information with little to no effort from the user with the goal of not being a burden that he or she must maintain. The mirror wouldn't be another activity, rather an enhancement to the already common use of mirrors in most modern bathrooms.

The mirror will do the thinking for the user. First, it will turn on and off by itself. Then, it will update with the user's calendar schedule, to-do lists, Twitter, news, and weather. The information wouldn't be thrown in the user's face, but unobtrusively displayed on the edges of the mirror to still allow use of the actual mirror. The use of touch-less gestures will keep things simple and easy to use. No keyboards to try to keep dry and clean. The gestures will also allow the user to still use the mirror despite whether their hands are wet or dirty. The mirror provides common information most people check their smartphones or tablets for, such as weather, news, Twitter and schedules. This allows the users to read, think, and plan their day while getting ready in the morning or night. The mirror has to be fun as well. It will provide music playback that can be controlled by their voice so there is no need for a mouse or keyboard.

Finally, the mirror must be smart enough to protect itself from the wet and humid conditions that occur in every bathroom. It will feature a humidity protection system where it will monitor the temperature and humidity levels near the hardware. If the temperature or humidity levels are out of the safe operating range, then a failsafe system will notify the PC system so it can shut off to prevent damage.

2.3 Requirements and Specifications

The smart mirror requirements and specifications took inspiration from people's everyday devices that they use including PCs, tablets, and smartphones. The mirror integrated similar features from each to give the user what they would expect out of a modern "smart" device. The following requirements and specifications were designed to satisfy the goals and objectives discussed in the previous documentation section.

2.3.1 PC Specifications

The smart mirror was run through a central computer that took in the data from the various sensors and peripherals which in turn were used to provide the user with its key functionality. The general hardware specifications for this control computer are provided in Table 2.3.1.1 below.

Motherboard	mini ITX Intel
RAM	4GB
CPU	Intel i3
Solid-State Drive	64GB
Power Supply	380 watt
Wireless Adapter	802.11b/g/n

Table 2.3.1.1 – PC Hardware Specifications

2.3.2 Necessary Features

The "necessary" features were requirements that are imperative to the project's design and objectives. They were designed and implemented before any additional features were worked on. The necessary features range from hardware requirements to interactive elements and to software elements. These necessary features are listed below.

- The smart mirror is designed to use a 32" diagonal display, positioned vertically, which will mounted behind a one-way mirror allowing only elements lit on the screen to be seen by the user.
- The smart mirror is interacted by the user through the use of an infrared/camera controller called "Leap Motion". This controller allows for touch-free multi-gesture recognition within an 8 cubic feet area above the controller.
- The smart mirror has a temperature and humidity monitoring system that will relay data to the user as well as a failsafe system to keep the electronics protected from water damage. The ideal operating range is 60 to 90 degrees Fahrenheit and between 20 to 80 percent humidity.
- The smart mirror contains speakers that allow for application notification sounds and music playback.
- The smart mirror user interface has a set of 7 standard applications that provide important information to the user including weather, Twitter, news, to-do list, calendar schedule, music, and clock.
- The smart mirror has an auto-on and auto-off system via use of a webcam mounted in the housing. The mirror will turn on when it recognizes that there is a user standing in front of the mirror. The mirror will turn-off after 2-minutes of no user present.
- The smart mirror has voice control through a voice recognition system developed into the user interface. The voice control allows for interaction with the music, to-do list, and twitter applications. The voice control is activated through a gesture provided by the user.

2.3.3 Additional Features

The "additional" features were requirements that are less imperative to the overall design and function of the project but would add extra depth and usefulness to the smart mirror. These were to be designed only once all the necessary features were implemented and polished to a suitable working state. These additional features are listed below.

- The smart mirror would make use of the digital zoom built into the webcam to provide more facial detail to the user on the display. This would be equivalent to a 10x zoom that could be used for putting on makeup or any other facial detail oriented task.
- The smart mirror would contain an outfit collection of the user's favorite clothing combinations. The user would take a photo of themselves in the outfit via the webcam and it would be stored in a sudo-database. This sudo-database would then be accessible to the user to identify what he/she wants to wear. Images could be displayed on one side of the screen to allow for direct comparisons to whatever the user is currently wearing.

3.0 Research related to Project Definition

3.1 Existing Similar Products and Projects

The projects and products similar to our smart mirror project cover a large spectrum of functionality and purposes. There were significantly more projects than actual products. Some blame can be put on the fact that the smart home is still an emerging market and is limited by the cost of manufacturing keeping the products out of reach from the everyday consumer. The fact that there were more projects shows the interest in developing a more affordable and functional smart mirror. Although, the actual products developed by a company delivered on features, they were either still in a development phase or already priced too high to be considered a viable competitor.

3.1.1 Projects

The following projects showed how the smart mirror can be designed in so many various ways. Each brought unique ideas and features to the term "smart". Not all of these projects were designed and built in the same year, so there is a noticeable difference in terms of use of available technology. Our smart mirror project has overlapping ideas with each of these projects but none of them are exactly the same. The projects researched are found below.

Interactive mirror is a touch and gesture functional mirror created by Alpay Kasal and Sam Ewen of Lit Studios [1]. The user touches the mirror, which has a built in touchscreen, to interactive with it. Unlike our smart mirror project, only one point of touch is recognized because it is emulating a mouse. Also, this mirror is less about data and more about artsy visuals. Users in the demo video show off different types of drawing and 2D games that are displayed using a projector. The fact that it emulates a mouse is nice because of the expandability and the range of functions capable. Yet, this still differs from our smart mirror since it isn't made to solve anything, only entertain.

The HUD mirror was designed by five students for a course at the Chalmers University of Technology in Sweden [2]. They used a two-way mirror to allow the LEDs they mounted behind to illuminate information through the mirror similar to how the smart

mirror will display information. This mirror was made for the bathroom and displayed time, weather, outside temperature, and a toothbrush timer by use of the LEDs. The toothbrush timer is actually a useful feature that our smart mirror project should consider. Also, instead of using a touchscreen for interaction, they used light dependent resistors (LDRs) as buttons behind the mirror. When "touched", the light changes and can perform a function specified in the arduino software. Despite being simpler, the HUD mirror has a lot of the same ideas as the smart mirror.

The magic mirror was developed by the New York Times Research and Development Lab [3]. It uses a TV with a mirror finish and uses a Microsoft Kinect to track movement and take in voice recognition. Also, it integrated a RFID reader to identify certain bathroom products. The whole system is run from a Windows PC just as the smart mirror will. The fact it can keep track of prescriptions and use the Kinect to "virtually" put clothes on the user are very inspiring features that given more time we'd love to integrate into the smart mirror. The magic mirror also allows the ability to check email, calendars, and social media, which confirms that our smart mirror will offer features that users are expecting.

3.1.2 Products

The smart mirror is definitely not a true consumer product yet. There are very few truly manufactured and ready for sale smart mirrors in the market. Those that are there are very different in terms of functionality, development, and price. It is certainly going to take a large smart home company to get behind this product and make it main stream to the consumer. Each product did have a common feature, which was health management such as weight. This is something our smart mirror doesn't have a direct focus on and maybe would be something to change in the design if our project were to go public. The products researched are found below.

The android-powered mirror created by the Japanese company, Seraku, uses an android table to power the mirror [6]. An LCD monitor covered with a semitransparent reflective glass is used to make the mirror. The mirror contains apps such as weather, news, weight, temperature, and water flow. Water flow is an interesting addition and could be excellent in helping water conservation. The mirror also uses RF proximity sensors to interact with the mirror with similar hands-free reasoning to why our smart mirror is using the Leap Motion controller. This mirror also comes paired with a scale, for analyzing and keeping track of the user's weight. The product isn't commercial ready so there is no estimate of price. A user's health is definitely the marketing point to actually selling interactive mirrors, despite manufacturing costs probably holding them back.

The Cybertecture Mirror is an actual product sold in Hong Kong [4]. It is made with a mirror screen and uses a cable TV similar remote, or smartphone, to navigate its applications. The mirror is equipped with wifi, two stereo speakers, fog resistant glass, and parts designed to operate even under bathroom humidity. The interface contains simple apps that allow for different user profiles. Apps include weather, social media, TV programs, virtual lighting, and health information. The health information is gathered

through a scale that is paired with the mirror. This mirror is truly a design and engineering goal for our smart mirror. It contains almost all the features of the smart mirror plus some extra. Though, the virtual vanity lighting seems to lack the true function of real lighting. Overall, an impressive mirror and shows that these type of mirrors are possible. The only drawback is that the launch price was $7,733 which limits its adoption in households to people who make a significant amount of money[5]. Our smart mirror project aims to be over 80% cheaper to produce.

3.2 Relevant Technologies

3.2.1 Gesture Control

One of the most important decisions in the design was what to use for interaction with the smart mirror since it directly influences the user and their experience. The elements in choosing which gesture control system to use depend upon what the user was expected to interact with on the screen. Also, the fact they are interacting with a mirror that ideally shouldn't need to be cleaned constantly because of finger prints. Option one for gesture control was using infrared camera sensors to track position and movement of the user's hand. Option two was to use a controller-based peripheral like a TV remote to select applications on the screen. Option three was to use a capacitive touchscreen with a mirror finish to take gesture input on the screen.

The use of infrared cameras and infrared lights is a common practice now for tracking motion. Most security systems use passive infrared sensors in their products to detect the presence of a person in the room or wherever the system is set up [9]. There are two popular gesture controllers that involve use of infrared camera sensors to decode motion, the Microsoft Kinect and the Leap Motion Controller. The Kinect has an infrared-light projector that blasts infrared light into the space it is facing and then the reflections are picked up by the infrared and RGB built-in cameras [8]. The Leap Motion is similar in that it uses two infrared cameras that pick up the infrared reflections that the three infrared LEDs output onto the user [7].

This use of IR cameras and IR emitters is visually demonstrated in Figure 3.2.1.1 below. The IR emitter covers the viewable area in infrared and then the IR camera collects the reflected infrared information. This data is then computed through vector analysis to get the depth and coordinate information on the object [36]. The Kinect and Leap Motion fulfill the objectives for the smart mirror project by being an easy-to-use, hands-free option for controlling the mirror's interface. The downside is that if the user is holding an object, the system might get confused. Also, the sensors won't work correctly if the camera lens gets dirty or blocked by something that obstructs its view of the user's hand. The final obstacle of using infrared light is that it can be interrupted by use of bathroom lighting that interferes with infrared light.

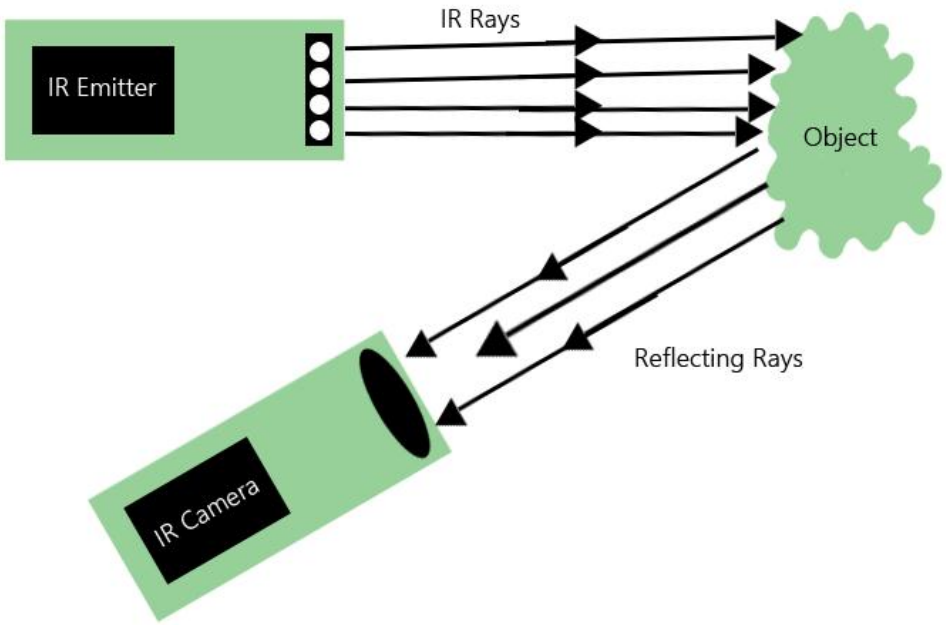

Figure 3.2.1.1 – IR Camera and Emitter – Recreated with data provided from Maximum PC

The use of controller-based controls is a slightly less hands-free option than the infrared sensors but they are commonly less buggy and are fairly easy to use. Two popular controller options are the Wii Remote and PlayStation Move Controller. The Wii remote uses an IR sensor bar and image sensor in the remote to track the position of the remote using PixArt's Multi-Object Tracking engine [10]. From this, the distance and direction of the remote is calculated. The PlayStation Move Controller uses a remote with a glowing sphere that is tracked through a camera in 3D space [11]. This would be able to use the webcam that is already designed to be part of the smart mirror. Each offers a viable option for wirelessly controlling the smart mirror with their pros and cons. The positive of using this method is that the tracking is more actuate and is less likely to misread the user. The downside is that the user must hold the controller and it violates the objective of interacting with the mirror in a hands-free manner.

The use of a capacitive touchscreen monitor instead of a TV is the third option. This would allow the user to physically touch and swipe on the mirror to interact with the interface. Like the TV, this touchscreen monitor would be covered with the mirror finish. The touchscreen would have to be tested to make sure the mirror finish didn't prevent the capacitive touch ability from working correctly. No cameras or IR sensors are needed, since the touchscreen uses the electrostatic field between the screen and the user to measure movement [13]. Assuming the touchscreen works correctly with the mirror, the interaction would be very exact and have the lowest "bug" rate in terms of user recognition. This method wouldn't involve any remote, but it wouldn't be a touch-free interface. The mirror could easily get smudged constantly which may hurt the user experience. Also, this option would be the most expensive as some of the best quality capacitive touchscreens cost over $3,500 [12].

Given these three options and the possibilities each bring to the table; the one the smart mirror project chose to go with is the Leap Motion controller. It came at a low price with high accuracy and a touch-less interface. It does have a higher risk of misreading the user, but programmed with simple gestures, they rarely occur. The Leap Motion fulfills the objectives mentioned earlier in this document and keeps the mirror clear of any fingerprint smudges.

3.2.2 Voice Control

Nowadays voice control is a very popular and useful technology that allows the user to perform certain functions by converting words and phrases into electric signals. Building entire speech recognition system is very time consuming and hence we integrated an already existing speech recognition system into our design. There were a few excellent voice recognition systems that we considered for our design such as Google Voice, AT&T Speech API, Nuance Voice Control SDK, and CMU Sphinx toolkit. The most important factors determined the speech system to go with was ease of integration, cost, and accuracy. The table 3.2.2.1 below compares different types of voice recognition software.

	Google Voice	**AT&T Speech API**	**Nuance Voice Control SDK**	**CMU Sphinx Toolkit**
Developer Price	Free	Free	$199	Free
Speaker Independent	Yes	Yes	Yes	Yes
Processing	Server	Server	Server/Local	Local
Audio File Type	Flac	Wav	Wav	Wav/Mp3
API Available	No	Yes	Yes	No
Limitations	Not released to developers	Requires oAuth2.0 server communication	Little documentation for C++ and Windows 8	Little documentation for C++
Language Requirement	Any	Any	Any	Any

Table 3.2.2.1 – Voice Software Comparison – Data provided from [71],[73],[74],[75]

Based on careful analysis of different parameters, we decided to choose AT&T Speech API since it as free, didn't require addition local processing, and allowed for Wav audio file support which the Windows 8 namespace is capable of handling.

3.2.3 Temperature Systems

There were seven basic types of temperature sensors that can be used for the design of the temperature regulation system. These seven types are resistive temperature devices, bimetallic devices, change of state devices, infrared sensors, liquid expansion devices,

thermocouples, and silicon diodes [44]. Each one of them has certain advantages and disadvantages and by analyzing all the trade-offs the best option will be chosen.

Resistive temperature devices use resistance values to display temperature. Its advantages include a wide temperature range from -400F to +1200F, high accuracy, long-term stability, and good linearity [45].

A Bimettallic sensor has two unlike metals connected together which will enlarge when heated and hence produce signals which is then converted into an electric signal. The advantages of using bimettalic sensors include independence from power supply and portability [44]. While some disadvantages are that they are less accurate than resistive temperature devices and thermocouples. Due to poor accuracy, the bimetallic sensors will not be used in the design.

Change of state devices change in color and/or appearance based on the temperature. Its advantages include low price, wide range of temperature values and a fast way of measuring. While its disadvantages are poor accuracy and slow response time. Due to poor accuracy, the change of state devices will be rules out as an option for this design as well.

Infrared sensors use radiation to measure temperature. Its advantages include the ability to measure temperature where it is difficult to use contact temperature sensor and long operating life. Its disadvantages are inaccuracy, high cost, and that it could be affected by other infrared radiation.

Thermocouples calculate temperature by measuring the change in voltage. The advantages of using thermocouples are that they are inexpensive and have a wide temperature range from -350F to 3200F [45]. However, the disadvantages are that it lacks long-term stability, low output sensitivity, and non-linearity.

Silicon diode sensors use forward voltage of the diode which depends on the temperature to measure the temperature [44]. The advantages of using silicon diodes are low cost, high stability, good sensitivity and great interchangeability. In addition, it can also be incorporated into an integrated circuit without affecting the cost significantly. It should also be noted that resistive temperature devices such as the thermistor require significantly greater power than silicon based temperature sensors and resistive temperature devices are more limited in terms of temperature ranges due to their nonlinearity.

Figure 3.2.3.1 below represents the relationship between voltage and temperature for thermocouple, RTD, thermistor, and I.C Sensors. As shown from the graph, only I.C. sensors have a linear response which is beneficial due to the fact that linearity implies an accurate response.

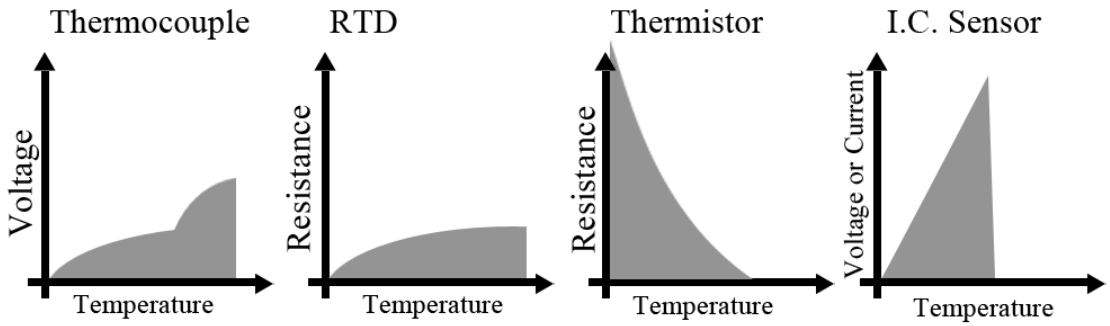

Figure 3.2.3.1 – Voltage and Temperature Relationships (Not to Scale) – Recreated with data provided from Louisiana Tech University

After thorough analysis of different types of temperature sensors, it was concluded that silicon based temperature sensors are the best fit for the design due to the low cost, high stability, excellent interchangeability, good sensitivity, linearity, and wide temperature ranges. The additional specifications used to make this design decision are listed in Table 3.2.3.1 below.

	Thermocouple	**RTD**	**Thermistor**	**I.C. Sensor**
Temperature Range	-350F ~ +3200F	-400F ~ +1200F	-100F ~ +500F	-40F ~ 257F
Interchange ability	Good	Excellent	Poor to fair	fair
Long-term Stability	Poor to fair	Good	Poor	Good
Accuracy	Medium	High	Medium	High
Repeatability	Poor to fair	Excellent	Fair to good	Excellent
Sensitivity (output)	Low	Medium	Very high	High
Response	Medium to fast	Medium	Medium to fast	fast
Linearity	Fair	Good	Poor	Excellent
Self-Heating	No	Very low to low	High	Low
Point (end) Sensitive	Excellent	Fair	Good	Excellent
Size Packaging	Small to large	Medium to small	Small to medium	small
Cost	low	low	low	low

Table 3.2.3.1 – Comparison of Temperature Sensors – Data Provided from Engineering Toolbox[45]

Based on the data from the table 3.2.3.2 below, I.C. temperature sensors and thermistors fit our design requirements better. Their accuracy in combination with the other elements was the reason for this choice.

	I.C. Sensor	**Thermistor**
Additional circuitry required	no	yes
Low power dissipation	Good	Poor
Batch to batch dependency	no	yes
Accuracy	Excellent	Excellent
Low supply current	Yes	no

Table 3.2.3.2 – Comparison of I.C.Sensor vs Thermistor – Data provided from "Tiny Temperature Sensors for Portable Systems" Texas Instruments Document [46]

The graph below in Figure 3.2.3.2 describes the relationship between the output voltage and temperature. From the graph it can be seen that the response of thermistor is not linear as the response of I.C. sensor thus concluding that the response of thermistor will be less accurate than the response of I.C. temperature sensors [49]. In addition, nonlinearity may introduce unnecessary noise. We can linearize the response of thermistor by using resistive networks but that would increase the complexity and cost. [47]

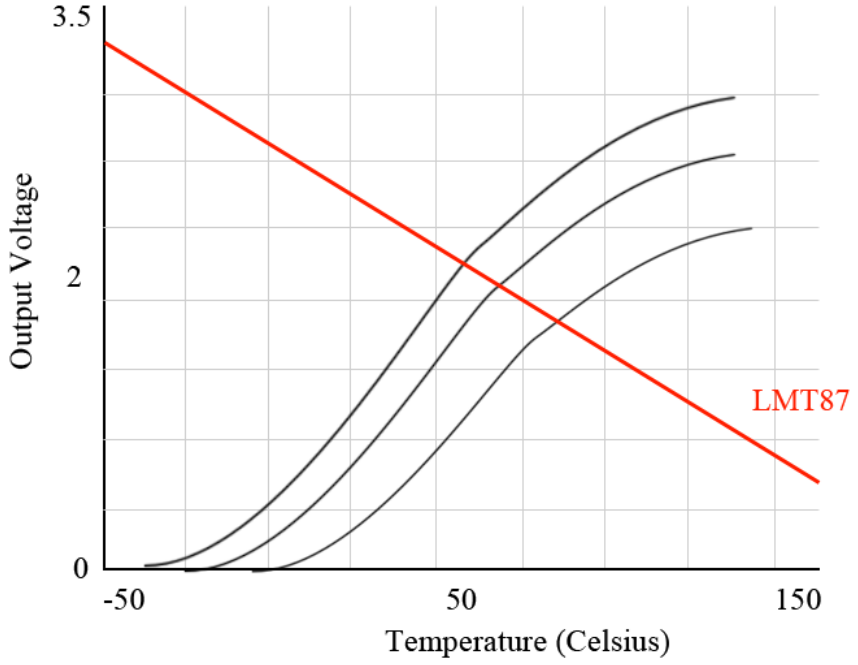

Figure 3.2.3.2 – Output Voltage vs Temperature – Recreated with data provided from ECNMag[47]

The picture below describes the relationship between supply current and temperature. As shown from the graph in Figure 3.2.3.3 below, the advantage of using IC temperatures is that they dissipate significantly less power because of the low supply current. When the temperature increases, the thermistor needs to supply more current which consumes more power and can also result in self-heating thus causing temperature errors [47].

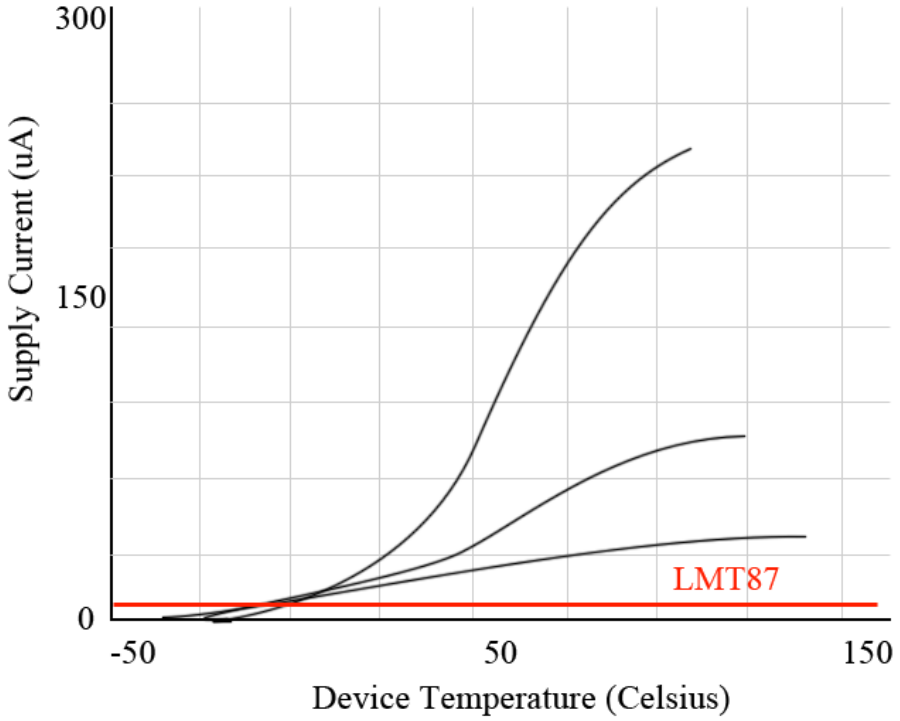

Figure 3.2.3.3 – Supply Current vs Temperature – Recreated with data provided from ECNMag[47]

Another important parameter that needed to be considered was the output impedance. The lower the output impedance is, the more accurate the response will be. The output impedance of thermistors tends to be higher and depends on temperature, while the output impedance of IC is usually low [48]. As a result, it is more preferable to use the IC sensor.

In conclusion, IC temperature sensors appeared to be the best option to use for our design because it had great accuracy, excellent linearity, long-term stability, wide temperature range, low power consumption, and a fast response time.

3.2.4 HDD vs SSD

Most components of the computer being used as the central control computer for the smart mirror have essentially only one type of technology. The only differences in RAM, CPUs, the motherboard, and PSUs are the specifications and speed. There is no actual

difference in the way the components are built or complete a task. The only part that currently has a division is regarding the secondary data storage. There is the most commonly used and oldest technology, the hard disk drive (HDD). The other option is the newer and growing storage method of flash memory in the form of solid-state drives (SSD). Factors involving the decision of which to use include price, speed, durability, and memory size.

The HDD has been around for over 50 years [15]. This has allowed for a significant amount of change and innovation in terms of performance, stability, and reliability. This has allowed for cost cutting materials to be discovered and used, making the memory size cost significantly lower than SSDs. By doing some quick research in terms of memory storage and price, it can found that for the same price as a 120GB SSD, a HDD with 1TB can be purchased [16][17]. That is a large difference in size. So clearly, HDD has advantages for cost and memory size. The next factor is speed. HDDs use drive motors to spin the magnetic platters and drive heads to read and write data. SSDs use flash memory chips which involves no moving mechanical parts. Having no moving parts allows the SSD to operate at a much higher read and write speed. Average access time for a SSD is about 0.1ms to 5ms while a HDD averages around 12ms [14]. Those numbers seem small and close, but when there are hundreds of reads and writes back-to-back, that difference adds up. The difference in read speeds can be seen even easier when files are fragmented over different sections of memory. HDDs read slower when on the inner rim of the platters whereas the SSD reads are consistent no matter where the data is [14]. This can be observed through Figure 3.2.4.1 below.

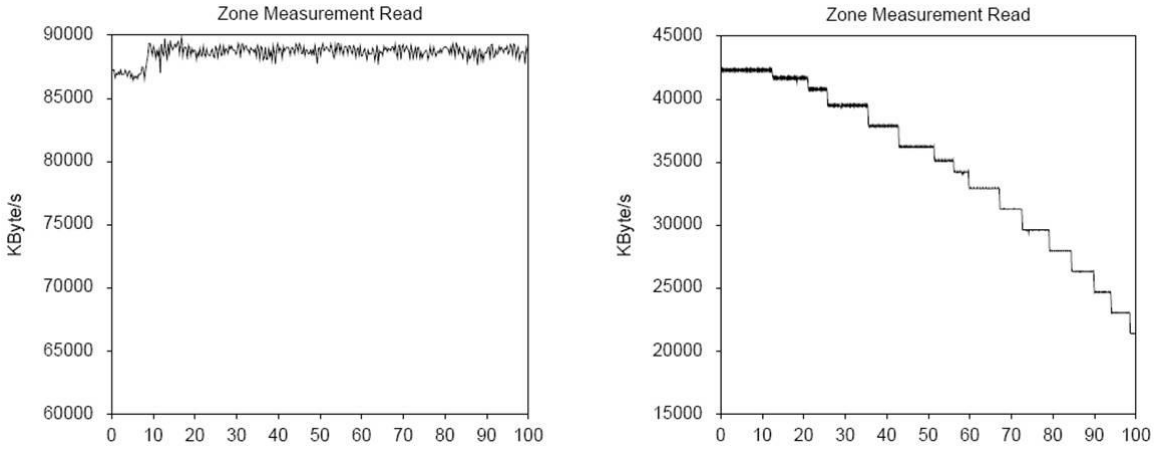

Figure 3.2.4.1 – Left – SSD Reads across the entire drive (Left – Outermost sector, Right – Innermost sector). Right - HDD Reads across the entire drive (Left – Outer rim, Right – Inner rim) Reprinted with permission from National Instruments [14]

It can be concluded that SSD have the advantage in terms of data speeds. The final factor, durability, refers to the ability to operate in more extreme temperatures for this smart mirror project. Various manufacturers give different operating temperatures for their products versus their competition but the overall conclusion is SSDs do better in extreme temperatures. On average, SSDs can operate from around 0 to 70 degrees Celsius. HDDs, due to the mechanical parts, have a smaller operating range of around 5 to 55 degrees

Celsius. The important aspect to look at is the humidity endurance, since this will be located in a bathroom setting. The SSD can operate in up to 95% humidity while the HDD has a threshold of up to 90%. There is also not a minimum humidity percent to optimize SSDs while HDDs ideally should have 5% or more of humidity [14]. With environmental operating endurance being a priority for the smart mirror project along with little need a high quantity of memory, the SSD was chosen to be the main secondary data storage for the control PC.

3.2.5 High Definition Display

For the High Definition Display, the smart mirror project planned to use a 32" high definition television. There were many different choices among televisions to choose from. Cost will play a factor in choosing a television. Another decision that has to be made is if we should use a 1080p vs. 720p and how much of a difference would it make on our smart mirror. The decision involved a choice between LED, LCD and Plasma. The television was behind a "two way mirror" that was glued to the screen. This television was used to project an image of certain applications through the mirror while the left of the mirror will still reflect the image of its surroundings. For the mirror to be able to reflect the television behind it must not emit any light thus being "dark." For the mirror to project the image behind it the television must be very bright. So the television we chose in turn must be very bright when the picture is shown and also while it is still on it must be very dark when needed. This made the mirror reflect the image when needed and also show the applications when needed as well.

After comparing between a 1080p and 720p resolution based on Figure 3.2.5.1 below, it was pretty evident that 1080p will produce a slightly clearer picture for our smart mirror [19]. Due to the fact that we do not need an extremely high quality video because we weren't watching movies or playing video games on the mirror, there was no need to spend any extra money on a 1080p television.

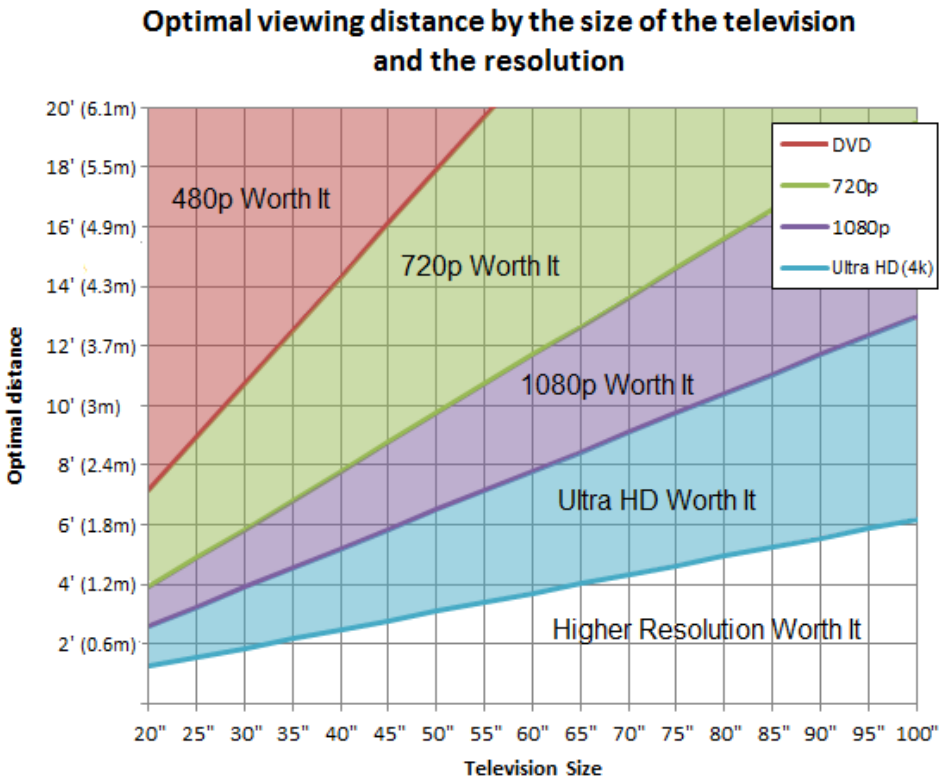

Figure 3.2.5.1 – Resolution Comparison Graph – Reprinted with permission from Rtings[52]

Regarding the decision between LED, LCD, and Plasma, there was no conclusive evidence on an advantage of any of these types of TV's with respect to the way it is being applied in the "Smart Mirror." Since there was no advantage between LED, LCD, and Plasma the television that was the most cost effective will be used. The television selected for the smart mirror project was an Insignia 32" LED TV that is 720p. The price and light weight, at 11.9 pounds, were the reasons for the selection of this television.

3.2.6 Humidity Sensor

There are many important parameters that needed to be analyzed when choosing what type of humidity sensor is needed for the smart mirror. Size, accuracy, cost, long-term stability, and many others were taken into consideration in this process. The most common types of humidity sensors are capacitive and resistive sensors. These are compared in Table 3.2.6.1 below.

	Capacitive	**Resistive**
Cost	low	low
Size	small	small
Accuracy	+/-1% RH	+/-2% RH
Long-term Stability	yes	yes
Response Time	30 to 60 s for a 63% RH step change	10 to 30 s for a 63% step change
Operating Temperature	-40C ~ 100C	-40C ~ 140C
Uncertainty	+/-2% RH from 5% to 95% RH with two-point calibration	-
Linearity	yes	no
Sensitivity	+/-2%@55%RH	4pf/%RH
Humidity Range	0 ~ 100% RH	5 ~ 95% RH
Output	-	Analog/Ratiometric
Votlage Supply	100 mV	2.3 – 5.5 V
Resistance to chemical vapors	yes	no
Resistance to temperature fluctuations	yes	no

Table 3.2.6.1 – Comparison of Humidity Sensors – Data provided from Digikey [50]

One of the crucial parameters that needed to be considered is the linearity of the response. Figure 3.2.6.1 below shows the relationship between capacitance and relative humidity in percent (RH%). As shown from the graph, the relationship between these two is almost linear which is good since it permits to provide an accurate response.

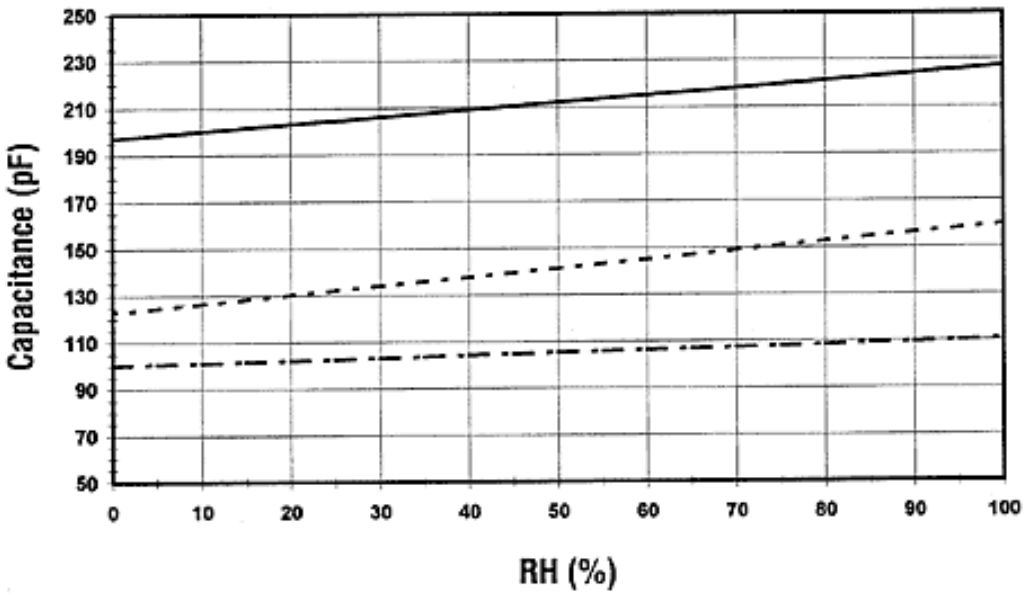

Figure 3.2.6.1 – Relationship of Capacitance and Relative Humidity – Reprinted with permission from Questex Media [51]

Figure 3.2.6.2 below illustrates the relationship between resistance versus relative humidity in percent (RH%). As shown from the graph, the relationship between these two parameters is nonlinear, which was undesirable due to the fact that it worsens the accuracy and can also introduce unnecessary noise.

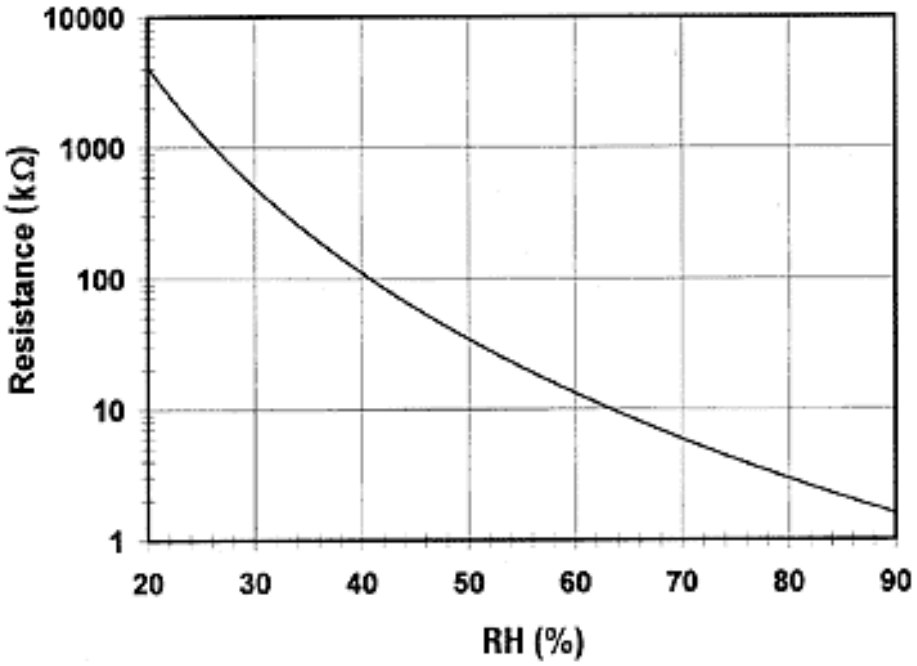

Figure 3.2.6.2 – Relationship of Resistance and Relative Humidity - Reprinted with permission from Questex Media [51]

After analyzing different specifications of capacitive and resistive humidity sensors, capacitive humidity sensors were chosen due to the fact that they require smaller voltage supply, hence consuming less power. In addition, it had near-linear response, resistive to chemical vapors, and a slightly higher humidity range.

3.2.7 MCU

The microcontroller was going to be used for one main application in the smart mirror. It was planned to be used to take the data from the humidity and temperature sensor and also have control signals for the temperature regulation system that communicate with the PC. There were many different microcontrollers that can be used to complete these tasks so the most efficient one was going to be used. The three microcontrollers being compared are shown below in Table 3.2.7.1.

	Arduino Uno	TI MSP430	Arduino Due
Clock Speed	16 MHz	8 MHz	84 MHz
Voltage	7-12 V	1.8-3.6 V	7-12 V
EEPROM	1 KB	4 KB	N/A
SRAM	2 KB	0.25KB	96 KB
Digital Pins	14	N/A	54
Analog Pins	6	14	12

Table 3.2.7.1 – MCU Specifications Comparison – Data from Arduino Website[41] and TI MSP430 Overview[53]

There were many advantages and disadvantages among the Arduino Uno, TI MPS430 and the Arduino Due. These 3 were chosen since they were reviewed as some of the easiest MCUs to program with websites providing multiple reference documents and several code samples. With time being a huge variable for a project of this size, saving time and effort contributed significantly for sitting to the milestones setup.

The first specification to compare was the clock speed. In a MCU, the clock speed affects how well the processor is at handling higher loads of data. As stated earlier, the MCU must handle two things. First is taking data from the humidity and temperature sensor. This required an onboard analog-to-digital converter (ADC) to convert the voltage levels outputted by the sensor s to a digital bit [53]. An ADC is on all three MCUs, so all met that criteria. Finally, the MCU must be able to regulate control signals to the temperature regulation system. This can be done with an analog or digital depending how the system is setup. This could also be completed by any of the three MCUs.

Now that the restrictions of the specific processes required of the MCU have been defined, further analysis could be completed. There were two limiting factors that will help decide on what MCU will be chosen for the project. The fact that all two of these tasks must be done, an above average processor load isn't required. This gave the edge to the Arduino Uno and MSP 430. The decision was then reduced to the Uno versus the 430. Both could perform the tasks needed but it came down to personal preference and with time efficiency in mind. This eliminated the MSP 430 as it wasn't preferred for the smart mirror's requirements, which left the Uno as the best choice for the project.

3.3 Strategic Components

3.3.1 Microcontroller

The microcontroller in this project was used for one main application in the smart mirror. The Arduino Uno microcontroller was chosen for the smart mirror for various reasons stated in section 3.2.7. The Uno, being a MCU, doesn't have one single function. The

16Mhz clock speed on the processor establishes the Uno as a mid-range MCU which allowed it to keep the benefits of slower MCUs but have some extra features. One of the best features was that it has a high voltage range, with being able to push its limits to 20V safely [41]. A high voltage broadens the spectrum of devices that can be run by the Uno.

Another great feature of the Uno was that it has several methods for communicating between devices. Such methods include USB serial, UART, I2C, and SPI [41]. Versatility was a good option to have in a microcontroller. The smart mirror project made good use of the serial communication for sending and receiving instructions from the central PC.

The multiple pins allow for expandability in the future. Only two pins were used for the smart mirror design but it was a useful option to have. Especially, the Uno had digital pins which a majority of MCUs don't have. This allowed for a backup option if the serial connection wasn't do-able for the project. There was no need to go through the analog pin and convert back to digital.

One of the most enticing features of the Uno was the software that is coupled with it. The instruction set to code programs has been designed to be simple and straight forward. A majority of other MCUs have powerful instruction sets but take significant practice to learn them. The C/C++ used in the Arduino software is fully documented and keeps commands visible to the user very simplified. For example, to simply read an I/O pin all that is required is "analogRead(pin number)" [41]. The software has eliminated the need to setup or turn-on any pins previously. The software is smart enough to do the work for the user which is an ideal situation for developers on a tight schedule. Hence, the Uno is perfect choice for the smart mirror project.

3.3.2 Leap Motion Controller

The Leap Motion controller was an essential component to the smart mirror project. This was the main interface that the user interacted with to use the smart mirror. The Leap Motion contains two IR cameras and three infrared LEDs that use infrared reflections to determine hand movement and individual fingers to translate to gestures [18]. The controller is only 0.1 pounds and 3 inches long. It is a USB peripheral and due to its small size, it can be placed within the bathroom environment discretely and unobtrusively.

The controller uses its cameras to translate movement and depth of every finger using a 150 degree side-to-side field of view. It also has a 120 degree field of depth view which can be seen in Figure 3.3.2.1 below. The arc going from back-left to front-right is the side-to-side 150 degree view. The bottom-left to back-right arc represents the 120 degree depth view. Both of those fields of view combined makes for 8 cubic feet of interactive area. Those movements get captured at 200 frames per second which then are interpreted using the Leap Motion API to recognize hands, fingers, and predefined gestures. As stated in the API documentation, there are four types of predefined gestures setup to be used for the developers. The first is a swipe movement that involves the hand, moving in a straight line, with fingers extended. The second is a circle movement where a circle is

"drawn" in the air using a single finger in either clockwise or counter-clockwise direction. The third gesture is a "screen tap" in which a single finger performs a forward tapping motion above the Leap Motion controller. The final gesture is a "key tap" and this is where a single finger performs a downward tapping motion above the controller. In order to interpret this, a minimum specification of a dual or quad core CPU, such as the Intel Core i3, was needed to decode the data and compute the mathematics involved in determining the object and location information [18].

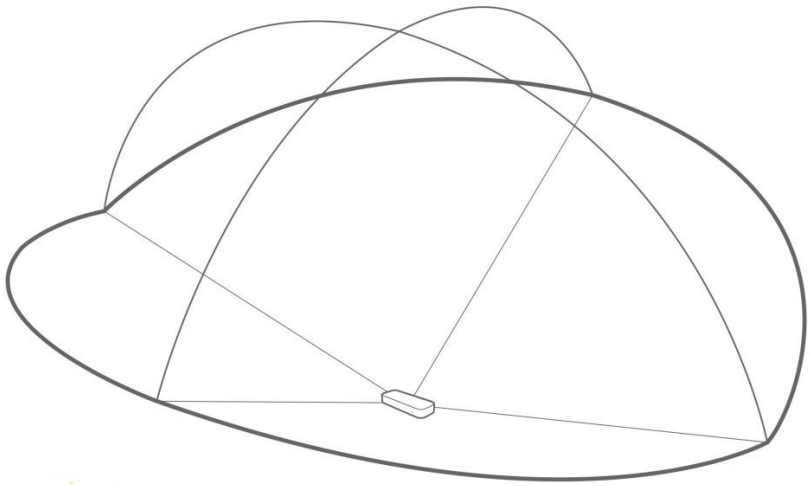

Figure 3.3.2.1 – Leap Motion Interaction Area - Reprinted with the permission from Leap Motion[18]

The Leap Motion controller allowed the user to navigate the smart mirror interface through simple points and swipes of the hand. The Leap Motion is primarily designed and optimized to look for hands and fingers within its viewing area. Using the product introduction visualizer software, it can be seen in Figure 3.3.2.2 below that the Leap Motion had identified the right-hand and each of its fingers. Within each finger, the joints have also been identified and emphasis on the finger tips has been applied. It also has interpreted the angle of the hand and wrist in its 3D space. As long as the Leap Motion cameras' lenses are kept clear of debris and excessive amounts of external infrared light, it will work despite the user having clean hands or not. This was beneficial in keeping the system easy and worry-free to use in a bathroom environment.

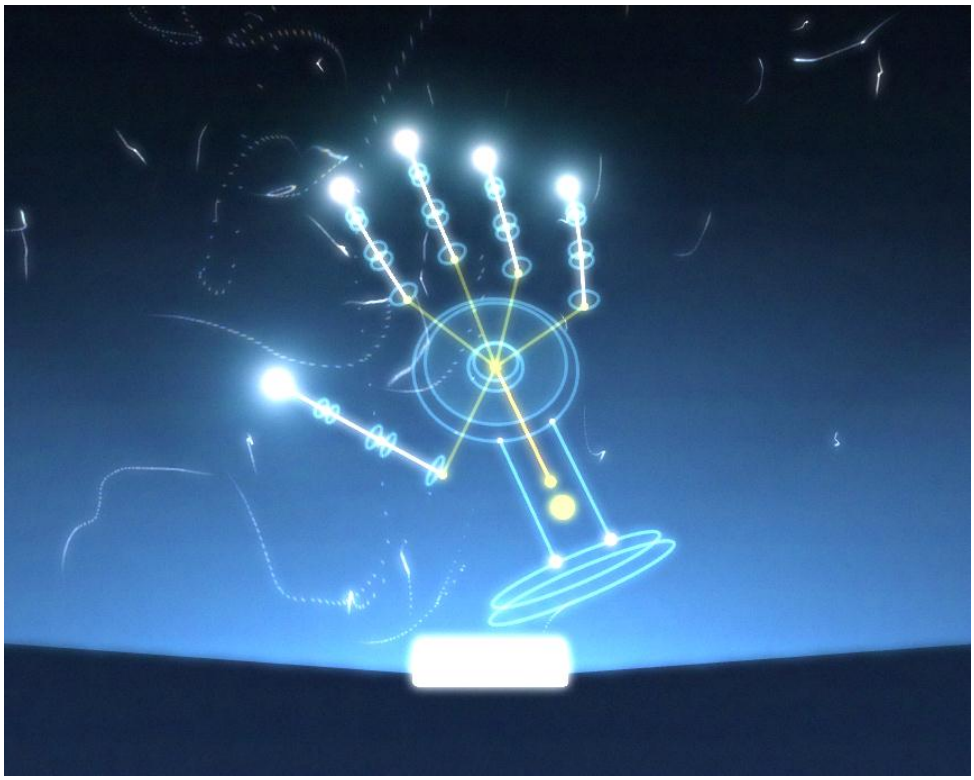

Figure 3.3.2.2 – Leap Motion Visualizer with hand – Data provided from Leap Motion[18]

3.3.3 Webcam

The webcam was another important component in the smart mirror project. The webcam provided both functionality and expandability. The main function of the webcam for the smart mirror was to detect the presence of user in front of the mirror. Once detected, the smart mirror will "wake up" from hibernating and display the applications on the screen. The webcam also determined when the user leaves, using a two minute waiting period, to then put the smart mirror back into hibernation in order to conserve power and resources.

The webcam that was chosen to use for the smart mirror is the Logitech C920 HD Pro Webcam. This webcam was selected for various reasons. The first was that the C920 is fully compatible with Windows 8, which is necessary since the main control program will be running on the Windows 8 operating system.

The next required feature was clarity. Clarity is achieved through full 1080p video recording support and on-device autofocus support [37]. This is essential because grainy, blurry image capture would complicate and possibly cause failure in terms of user detection, the main application of the webcam. Blurry pixels changing constantly due to a poor camera lens would result in a higher motion detection failure rate since such algorithms are based on frame-to-frame changes. Additional low-light correction provided with the C920 will also help achieve better image clarity.

Another added benefit of the C920 was the fact that it has H.264 video compression right on-board the device [37]. Having this compression done on the webcam's own chip reduces the workload of the control PC software and the memory space needed to store the frames.

Finally, given that time permits and that the additional features of the smart mirror were being aimed to be completed, the 15MB photo capability would have been necessary for the outfit "database" [37]. This "database" would store images of the user in their favorite outfits which they can use in the future for reference images on what to wear. For the user to have a clear and detailed image of their outfits is essential since blurry pictures would be of no use. Not only clarity, but details would be more defined using the C920 as well. Additionally, if the smart mirror were to expand to allow video conferencing, this webcam would be able to deliver the HD quality most users expect in this day and age.

3.3.4 Audio Capture

The Logitech C920 Webcam microphone was used for audio capturing. The chose to go with the built-in microphone of the webcam versus an external microphone was based on finances and whether an external microphone would do a better job. Since voice recognition accuracy is mainly based off of the software implementation, it was determined that it wasn't worth using another external peripheral to capture the audio instead of the built-in webcam microphone.

As stated earlier, the main purpose of the microphone was for the voice recognition system. The C920 webcam microphone has a dual-microphone setup which is beneficial over a traditional single-microphone setup. The advantage of having two microphones is that one handles the direct voice and the other attempts to filter out the ambient noises. The main microphone always receives the sounds more vividly and strongly, while the second microphone picks up the ambient noises around the same level as the other. The unwanted sounds from the second microphone are filtered from the primary microphone which thus leaves, mainly, the voice that needs to be recognized [42]. This principal of noise canceling can be observed in the visual representation below in Figure 3.3.4.1. The red lines are ambient noises and the blue line is the noise that is desired. The two microphones hear the same sounds, except the desired noise is stronger in microphone 1. Then the combination of data from these two microphones goes through a filter which outputs the desired noise.

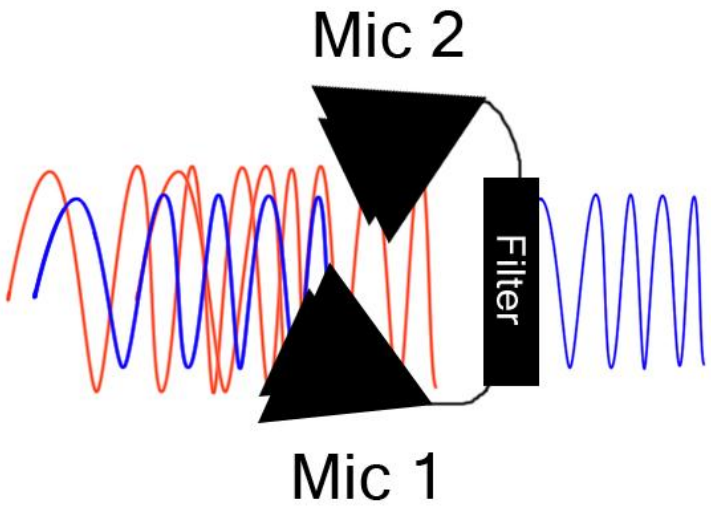

Figure 3.3.4.1 – Dual Microphone Noise Canceling (Red:Ambient Noise, Blue:Desired Noise)

3.3.5 Control Computer

The control computer was the main central part of the smart mirror project. All the data and control signals ran through the control computer. The control computer was essentially a compact PC. It ran the Windows 8.1 operating system where the designed Windows 8 application controlled the mirror and provided the GUI to the screen. The control computer was made up of a mini ITX Intel motherboard that had the modern standard of USB ports, audio ports, and video output such as HDMI. The graphics were rendered using integrated graphics on the Intel Haswell i3 processor. The motherboard also housed 4GB of DDR3 RAM as well as a 64GB solid-state drive. Finally, a 380 watt power supply powered the computer. A mini ITX computer case housed all the parts mentioned and provided appropriate air cooling to the electronics. All these parts were contained within the designed temperature regulation system to provide maximum protection against temperature and humidity damage to the electronics. The fail safes put into the design used the elements of the regulation system until they become ineffective in which at that point the system will be ordered to shut off.

The Windows 8 application that ran off this hardware was created using Microsoft Visual Studio 2013 and the Windows API. Since this software was the “hub” where all parts of the smart mirror interact, it was necessary for the motherboard to provide an appropriate number of USB ports for the peripherals. These included the Leap Motion controller, the webcam, and the MCU serial connection. The multi-core Intel i3 processor was required because of the minimum specification requirements of the Leap Motion controller as well as the Logitech webcam. The HDMI port was used to output the GUI to the HDTV. This provided easy compatibility with most current HDTVs and also outputs a clear high-resolution image.

3.3.6 Temperature Control

The smart mirror was designed with a temperature control PCB circuit that relayed temperature and humidity data through the MCU to the central PC. A comparator was used to regulate temperature of the entire computer system. The comparator was an op-amp which operates in open-loop configuration [54]. Comparators usually have a grounded emitter and an open collector output which allows the pull-up voltage source of the output to differ over a wide range, thus enabling comparators to interface with different logic circuits [54]. Because of the open-loop configuration, comparators usually do not have Miller compensation capacitors. As a consequence, the bandwidth is wide. Figure 3.3.6.1 below displays a voltage transfer characteristics of an open-loop comparator.

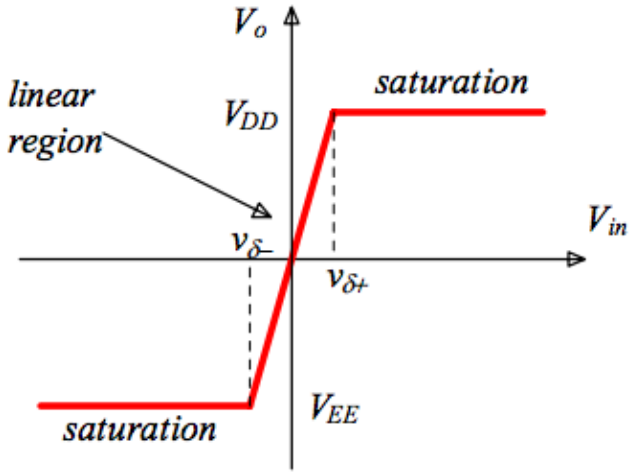

Figure 3.3.6.1 –Transfer Characteristics of Open Loop Comparator Voltage – Reprinted with permission from MIT Open Courseware [54]

The purpose of comparator was to compare two voltages and determine which one is larger. As shown from Figure 3.3.6.2 label "a" below, it is biased at voltages +Vs and –Vs. Figure 3.3.6.2 label "b" shows us that the output will be driven to a high saturated state V_H when V2 is a little bit greater than V1 and the output will be driven to a low saturated state V_L when V2 is a little bit smaller than V1. The transition region is between –δ and +δ. Frequency stability does not need to be taken into account due to the fact that the comparator operates in one of the two states: high and low. Additionally, it's not limited by the slew rate for the same reason. The response times for the output to alter states vary from 30 to 200ns [54]. Figure 3.3.6.2 below demonstrates two types of comparator circuits: inverting and noninverting.

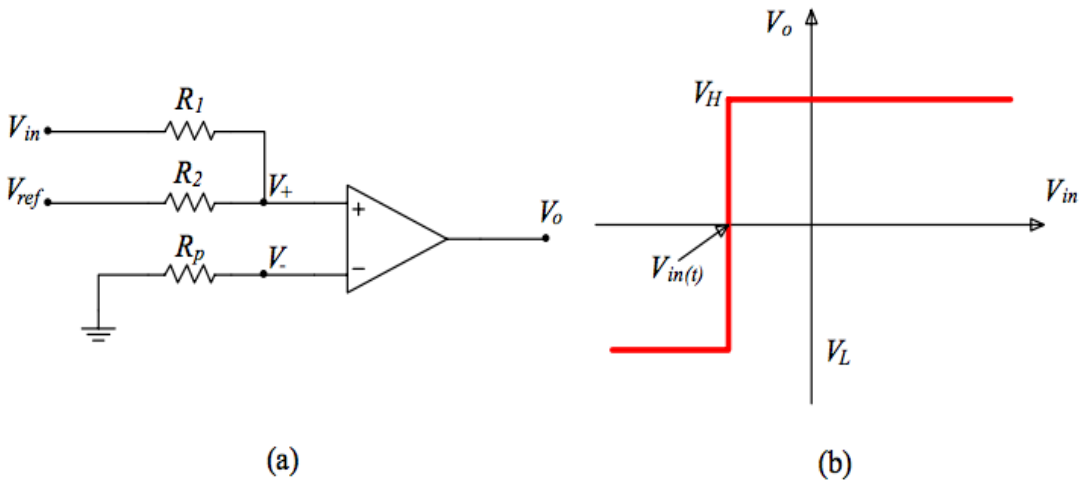

Figure 3.3.6.2 – Non-inverting Comparator: a) circuit b) voltage transfer characteristics – Reprinted with permission from MIT Open Courseware [54]

The circuit amplifies the difference between Vi and Vref and it outputs the result Vo. In the case when Vi is greater than Vref, the Vo goes to a high saturation state from a low saturation state. This only occurs if it's a noninverting comparator circuit. It goes to a low negative saturation level from a high saturation state if it's inverting comparator circuit [54]. In this case the relay should turn on when a certain temperature level is reached, a noninverting comparator circuit should be used. If we use superposition principal for Figure 3.3.6.2 label "a" above, then V+ = R2/(R1+R2)*Vref + R1/(R1+R2)*Vi. The crossover voltage, which is defined as the input voltage at which the output changes state, occurs when V+ = 0. Therefore, the new equation to calculate the required input voltage will be:

$$R2*Vref + R1*Vi = 0$$
$$Vi = -R2/R1*Vref$$

This meant that v0 is in high state when vi is greater than the crossover voltage. However, there is an important problem that needs to be addressed. If there is some noise in the system, it will get amplified by the open loop gain which causes the output to keep rising and falling, thus wasting power. Therefore, a Schmitt trigger is included to solve this issue. Schmitt trigger, also known as bistable multivibrator, produces bistable characteristics by using positive feedback [54]. It separates switching points so that the input is subjected to a big reversal before the inverse transition takes place. Figure 3.3.6.3 below illustrates the Schmitt trigger circuit, voltage transfer characteristics as input voltage increases and decreases, and net voltage transfer characteristics.

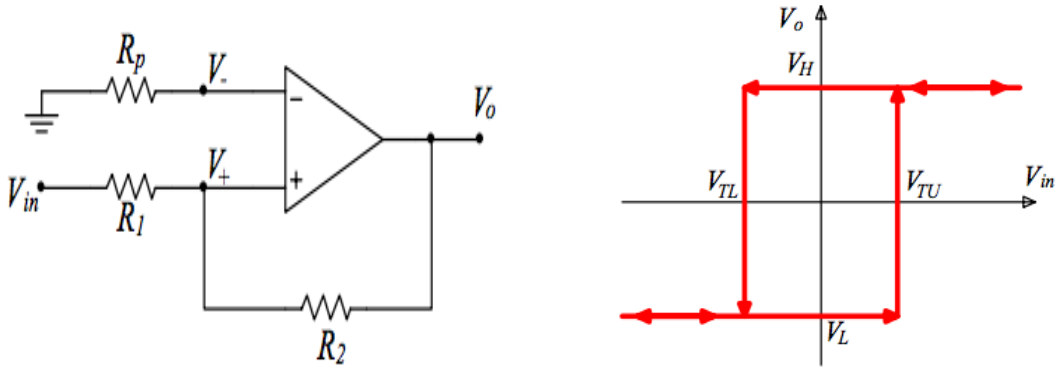

Figure 3.3.6.3 – Comparator Circuits: a) noninverting and b) inverting - Reprinted with permission from MIT Open Courseware [54]

We can see from the Figure 3.3.6.3 that V0 = VL (negative value) when Vi is negative and the output is in low state. The crossover voltage Vi becomes equal to Vth when V0 = VL and v+ = 0. However, the problem with Schmitt trigger circuits is that the open-loop saturation voltages of the comparator may not be accurate enough and can fluctuate from one comparator to another. If another limiter network is added, output saturation voltage will be able to be controlled and therefore increase the accuracy. The typical Schmitt trigger circuit is displayed below in Figure 3.3.6.4.

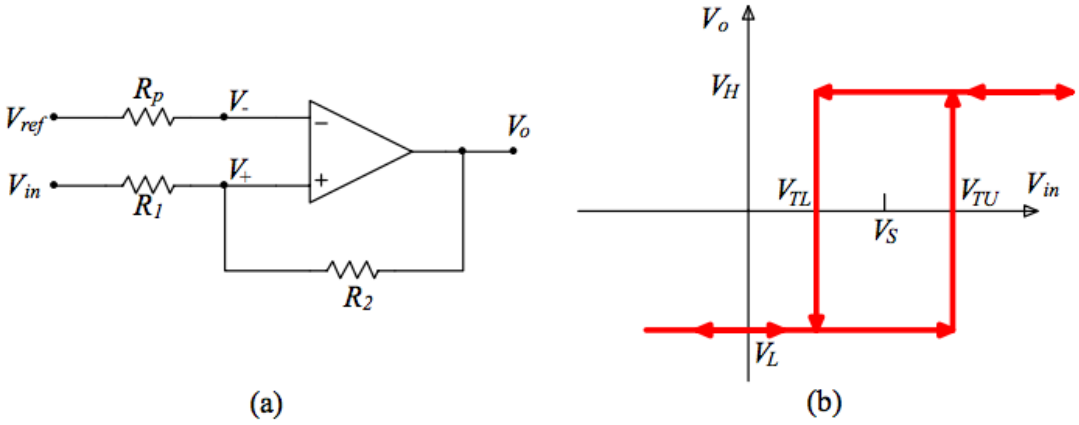

Figure 3.3.6.4 – a) Schmitt trigger circuit with reference voltage and b) voltage transfer characteristics - Reprinted with permission from MIT Open Courseware [54]

As we can see from Figure 3.3.6.4, Vtl and Vth depend on R1, R2, Vref, V_gamma (diode voltage) and by manipulating these values the region between Vtl and Vth can become as wide as we needed. Therefore, if the temperature fluctuates it will not cause any problem. This guaranteed stable and accurate temperature control in the smart mirror system.

3.3.7 Humidity Sensor

Two very common humidity sensors are P14 and MK33 developed by Innovative Sensor technology. Table 3.3.7.1 below compares technical characteristics of these two sensors:

	P-14	**MK33**
Humidity Operating Range	0 – 100%RH	0 – 100%RH
Operating Temperature Range	-50C – 150C	-40C – 190C
Low drift	yes	yes
High humidity stability	yes	yes
High chemical resistance	yes	yes
Linearity error	< 1.5% RH	< 2.0% RH
Frequency Range	1- 100 kHz	1 – 100 kHz
Sensitivity	0.25pf/%RH	0.45pf/%RH
Loss Factor	<0.01	<0.01
Hysteresis	< 1.5% RH	< 2.0% RH
Response time	< 5 s	< 6 s

Table 3.3.7.1 – Humidity Sensor Comparison –Data provided from Innovative Sensor Technology [55]

The figure below describes the relationship between temperature and relative humidity for P-14 humidity sensor. The space inside of the closed box refers to allowed humidity-temperature range in Figure 3.3.7.1 below.

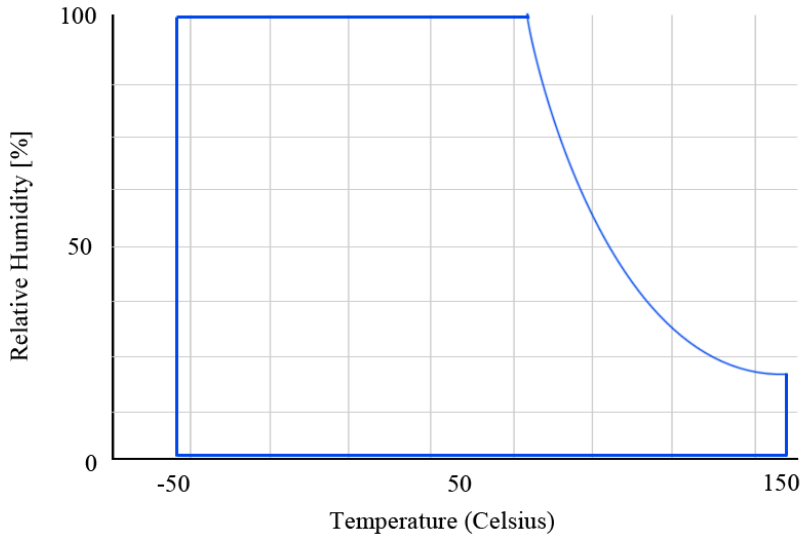

Figure 3.3.7.1– Temperature vs Relative Humidity (%) – Recreated with data provided from Innovative Sensor Technology [55]

Figure 3.3.7.2 below describes the relationship between temperature and relative humidity for MK33 humidity sensor. The space inside of the closed box refers to allowed humidity-temperature range.

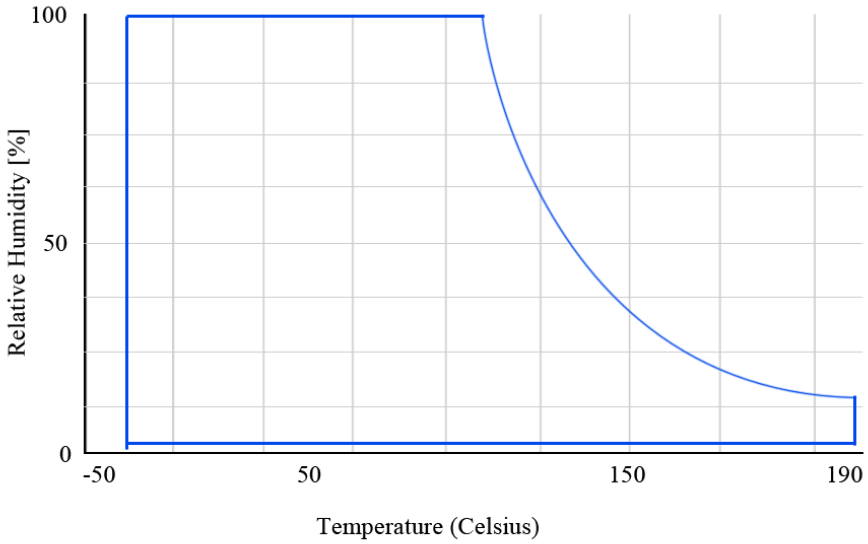

Figure 3.3.7.2– Temperature vs Relative Humidity (%) – Recreated with data provided from Innovative Sensor Technology [55]

As can be seen from these two graphs, MK33 has wider humidity-temperature range. In conclusion, it was better to use P-14 than MK33 because its linearity error is smaller, it has a better sensitivity, and its response time is faster.

3.4 Possible Architectures and Related Diagrams

3.4.1 Compiler IDE

The integrated development environment (IDE) is the program used to develop applications for a software projects. Not all IDEs are the same, they vary in terms of what tools and features are offered. Many include support for multiple programming languages along with the ability to maintain an easily readable file hierarchy. The IDE chosen to be used for the smart mirror project was Microsoft Visual Studio 2013. This IDE is not only the recommended program to use to create Windows 8 applications but has the added bonus of being free for university engineering students through Microsoft DreamSpark.

Microsoft Visual Studio has support for several different programming languages, such as visual C#, visual C++, HTML, and visual basic [38]. This provides a lot of flexibility on how the project can be approached. For this project, the primary language was visual C++ coupled with XAML for the visual user interface. XAML is a “declarative markup language” which essentially makes for creating a UI faster and easier [39]. The best feature about XAML is that the UI can be designed and coded separately from the logic

portion of the application. This prevents time consuming changes if something were to be altered within the UI or the logic portion of the application. Visual Studio provides two views for the XAML, one in syntax form and one in the GUI form. Users can edit one or the other to achieve their visual vision. The GUI allows for "drag and drop" of elements such as buttons to the output template. These elements can then be altered through code or by modifying the list of properties that Visual Studio lists for them.

Like most IDEs, Visual Studio offers a code editing interface where the code is written. Cosmetic and functional touches such as text highlighting, auto-completion, hierarchy traversal, and more are offered to increase productivity and to prevent mistakes when coding. Since it is designed to build Windows applications, the needed APIs are built in for easy use. The IDE also contains the appropriate compiler to build and debug the code written [38]. This allows code to run instantly and test for correctness and bugs. The provided console window lists warnings and errors regarding the code written.

Visual Studio specifically benefits the smart mirror projects in various ways. First off, since the project was planned to be run through a Windows 8 application because of the Windows 8 operating system, visual studio was the obvious choice because of the advantage it has in developing such applications. Visual Studio also eliminated the need to use a 3rd-party framework to develop the GUI since the XAML provides all the functionality that will be needed as a developer and for the user. Having access to the Windows API allowed easy integration of the various peripherals including the webcam, Leap Motion, and MCU. Finally, the functional touches built into the IDE kept development efficient since sticking to the schedule is a priority.

3.4.2 Code Libraries

When programming, there is bound to be multiple code libraries being used. Libraries provide access to functions, architectures, and algorithms specifically designed for their platform. Without code libraries, those included inside APIs, programming would be near impossible for programming on a new system. The five most prominent APIs used in the smart mirror project are the following:

- Leap Motion API
- Windows 8 Store API
- AT&T Speech API
- Toodledo API
- Twitter API

The first API, Leap Motion, contains the code to control and manipulate the Leap Motion controller. The API contains all the classes and their functions that allow for the motion processing to be done correctly and be stored into useable variables. The main classes being used in this project are the ones relating to their pre-programmed gestures. The gesture class contains all the information on their swipe, circle, screen-tap, and key-tap gestures [18]. The functions within the class contain information such as the direction, position, finger count, and various other data. This is what will be used to make the UI controls for the smart mirror.

The second API is the Windows 8 store API. This is a huge set of libraries due to the fact that not only does it contain the new XAML code libraries involved in the Windows store applications but also most of the Win32 libraries that are used to create all Windows applications. The XAML libraries will be used to generate the GUI and connect to logic code running the UI [39]. Specific libraries such as the ones relating to serial data transfers, webcam pixel data, and HTTP connections will be the focus for the smart mirror project. Serial data is needed to link the central PC and the MCU. The webcam data access will be needed to get image information to run the motion detection algorithm. The MediaCapture namespace was used for "speaking" with the webcam. Finally, the HTTP connections will be needed to set up data links for the applications that use external server data such as the weather and news apps.

The third library source is the AT&T Speech API. This documentation provides the needed information for authorization and communication with AT&T's speech server. They use the OAuth 2.0 framework which makes use of access tokens to send and get requests from their server. Upon setup of a developer account, the access token is obtained through a HTTP Post request with the appropriate authorization bearer [71]. To get the voice translated to text, the audio file is uploaded in a Post request to their server and then, in JSON format, the text is returned. Then the translated text is handled by the program to execute the necessary command.

The fourth API is the Toodledo API. This documentation provides the steps to register the app, obtain an authorization token, and then the appropriate tokens for subsequent calls to their server. Like the AT&T Speech API, Toodledo also uses the OAuth 2.0 framework. The OAuth 2.0 flow from initial sign-in to getting or adding tasks can be seen in Figure 3.4.2.1 below. Using the authorization broker namespace from the Windows 8, the user is prompted to login. From here, an authorization code is returned. This code is sent back to Toodledo's server to get an access token. The access token is used in the requests to upload and download new tasks from the user's profile. The use of HTTP post request passes the appropriate data to the server. That information is returned in a JSON format. The access token expires every 2 hours so a refresh token has to be used to obtain a new access token. The refresh token expires every 30 days and a new one is returned whenever a new access token is obtained.

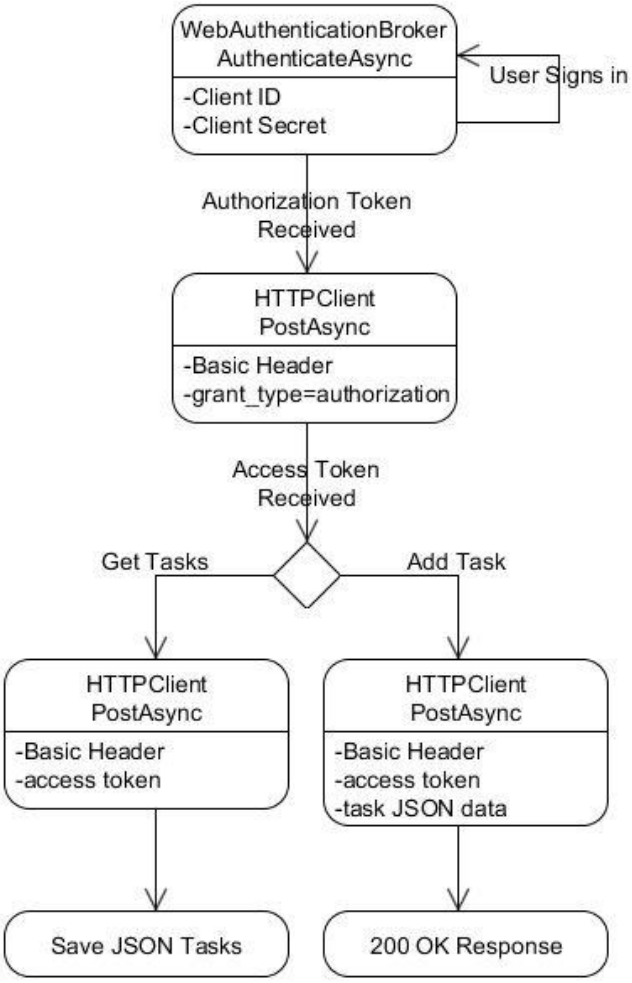

Figure 3.4.2.1 – Toodledo OAuth 2.0 Basic Flow

The fifth API is the Twitter API. This API documents how to use the OAuth 1.0 framework to authenticate and communicate with Twitter's secure servers. Like Toodledo, the user is first prompted to sign in using the authorization broker namespace. Then a series of token handshakes are performed making use of a nonce, timestamp, and signature. The nonce is a token generated by the code on each request that is a series of base64 encoded random data for security purposes. For the same purpose, a timestamp is created that represents the number of seconds from the Unix epoch. Finally, the signature is created using method "HMAC-SHA1". The Windows 8 namespace has a MacAlgorithmProvider which has that security method built-in for semi-easy key creation. The signature is made of several pieces of data including the timestamp, client login, client secret, nonce, and signing key. This process is more complicated than the newer OAuth 2.0 framework. Once all setup, simple HTTP post requests obtain the latest tweets from the user's timeline as well as allow the app to post new tweets with use of the voice recognition.

4.0 Project Hardware and Software Design Details

4.1 Project Block Diagrams

To get an overview and better understanding for the smart mirror project, block diagrams were drawn out to be able to see the "whole picture". Since this project involves a significant amount of software, both a hardware and software block diagram were created. The block diagrams provide a top level view on how each part of the system interacts.

4.1.1 Hardware Block Diagram

The blocks in the hardware block diagram or system block diagram for this project represent an important entity. Figure 4.1.1.1 below shows how all the components relay through the central pc, which is the main driver for the smart mirror. The arrows show how each block is interacting with the next. Most blocks are either an input or output in this diagram.

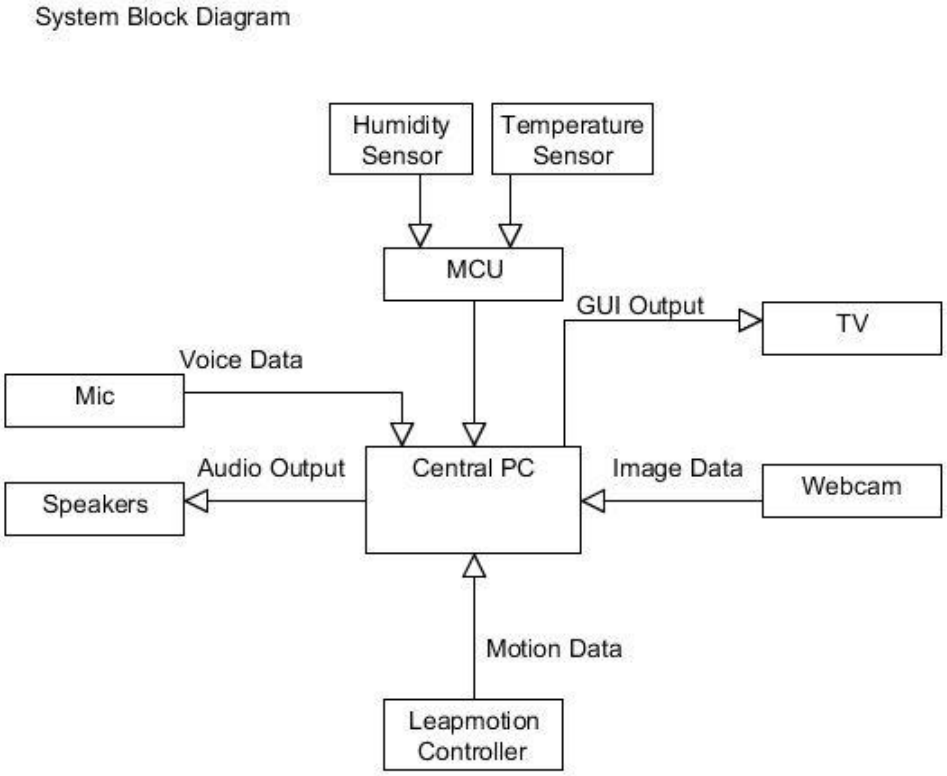

Figure 4.1.1.1 – System Block Diagram

4.1.2 Software Block Diagram

The software block diagram is a detailed abstraction of the two main processing units in the smart mirror, the MCU and the PC. As seen in Figure 4.1.2.1 below, the MCU will take care of all the temperature data processing that is used in the humidity regulation system. The MCU then sends the data to the PC. The central PC takes care of the user gesture input, webcam image processing, voice processing, and outputting the GUI to the user. From the block diagram, it can be seen, for example, that the central PC needs a hefty processor and appropriate optimization to handle all the data being manipulated within it.

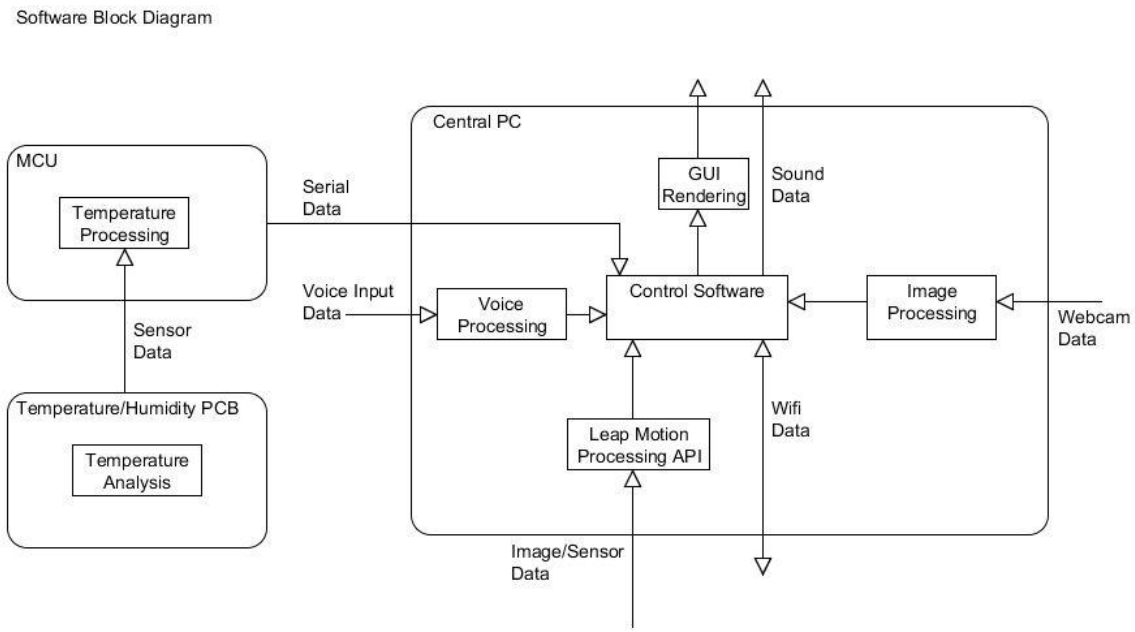

Figure 4.1.2.1 – Software block diagram involving MCU and PC

4.2 Gesture Control Subsystem

4.2.1 Leap Motion Specifications

The Leap Motion is a compact IR camera controller, which allows for the interpretation of hand and arm movements by use of infrared light reflections off the user. The Leap Motion makes use of two IR cameras and three infrared LEDs that cover the hand in infrared, which then gets recorded by the camera [18]. The camera records at a high frame rate the data it is receiving and then sends that data to be processed by the API. This processing involves complex math, which in turn demands for a high processor and memory requirement. Table 4.2.1.1 below shows the system requirements as well as the physical specifications of the controller. As can be seen, an above average computer system must be provided to allow the Leap Motion to function correctly and as specified.

Weight	0.1 pounds
Dimensions	0.5 inches x 1.2 inches x 3 inches
CPU (Min. Requirement)	AMD Phenom II or Intel Core i3
Memory (Min. Requirement)	2 GB
Operating System (Min. Requirement)	Windows 7 or 8

Table 4.2.1.1 – Leap Motion Specifications – Data provided from Leap Motion [18]

The Leap Motion "interaction" area is limited by the viewing angle of the infrared cameras used. As can be seen in Table 4.2.1.2 below, the front and back viewing angle is less than the left and right viewing angle. Therefore, the interaction in terms of depth is more limited than the left and right sides of the controller. For each though, the max distance is 2 feet. Although the side-to-side distances aren't a concern, the max height distance being 2 feet won't be enough to completely "cover" the mirror in a viewable area. Therefore, compensations must be made in the code to allow for full mirror interaction with only a 2-foot interaction height available to the user.

Controller Side	**Viewing Angle**
Front/Back	120 degrees
Left/Right	150 degrees

Table 4.2.1.2 – Leap Motion Viewing Angles – Data provided from Leap Motion [18]

4.2.2 Supported Gestures

The gestures supported are expansive since the API allows for customization of individual and unique gestures. Unfortunately, due to time constraints, the main gestures used for the smart mirror will make use of the gesture subclasses within the gesture class in the Leap Motion C++ API. As stated earlier, the gesture list contains four gestures and one invalid gesture state. The four gestures are swipe, circle, screen-tap, and key-tap.

The first gesture, swipe uses the SwipeGesture class within the Leap Motion API. Besides the inherited functions from the Gesture class, there are specific functions that make the swipe gesture customizable to whatever motion is necessary. The five specific functions included are:

- Direction
- Pointable
- Position
- Speed
- Startposition

The first one, direction, is unit direction vector that contains the data to determine the direction of the swipe. This vector contains x and y values that can be compared to determine the relative angle that the swipe is moving. So the x and y values can be used to see whether the swipe is mostly horizontal, vertical, or something at an angle. These values can be adapted to have a varying tolerance if the swipe that it is searching for is abnormal.

The second one, pointable, is straightforward in that it represents which finger is performing the operation. This allows for customization in which certain fingers are searched for to make it a valid motion.

The third function, position, contains the vector unit that represents where the swipe is currently occurring within the frame of view of the Leap Motion. This means, it contains the information for the position within the 8 cubic feet of viewing space that the Leap Motion is capable of seeing. The data is represented in millimeters for extreme accuracy.

The fourth function, speed, is how fast the motion is being performed. This is measured in millimeters per second and can be placed as a restriction to make the gesture more exact. Speed restrictions can prevent unwanted swipes by accident and prevent frustration from the user.

The final function, startPosition, is the origin vector of where the swipe began. This is measured in millimeters within the Leap Motion field of view and allows for custom start positions. Therefore, a gesture can perform different actions depending on where the swipe began. For example, the left most quadrants can perform something different from the same swipe in the right most quadrants.

The second gesture is the circle gesture. Similar to the swipe gesture, the circle gesture contains it own functions within the API that can be used to allow for individual circle gesture customization. The functions are as follows:

- Center
- Normal
- Pointable
- Progress
- Radius

The first function, center, is the recognized center point of the circle within the field of view. The unit vector is measured in millimeters from the Leap Motion origin in its frame of references. This allows for customization of where the circle must originate in order for it to be recognized as a viable gesture within the program.

The second function is the normal. The normal is a vector that relates to the circle being traced. When drawing the circle in a clockwise motion, the normal vector points in the same direction as the drawing object. This is opposite when drawing a counter-clockwise circle. This data is all essentially determined by the angle between the normal and the pointable object.

The third function is the pointable. This is the same as it was with the swipe gesture and represents the finger that is creating the circle. Related to the pointable is the fourth function, progress that counts the number of times the circle has been drawn. So using the progress function, a value is returned that represents how many revolutions the finger has made. For example, a value of 2.5 means that 2 full circles and one half circle was completed.

The final function, radius is simply the value that represents the radius of the drawn circle. This radius is in millimeters and can restrict the gesture recognition to larger or smaller circles depending on what is set in the code.

The third and fourth gestures are very similar. They are the screen-tap and key-tap. The screen-tap is more of a forward, pointing motion while the key-tap is a downward, pointing motion. Each of these gestures has the same functions for each of their classes. These functions include the following:

- Direction
- Pointable
- Position
- Progress

The first function, direction, is a unit direction vector that represents where the finger is moving. This looks for the specifics of each screen-tap and key-tap to verify the gesture is occurring. It is possible for the vector to be zero when there is no movement from the finger. This vector allows for the specific angle and "pointing direction" to be interpreted. So, for example, a finger pointing slightly to the left can be read as pointing at an object on the left despite the finger being in the center region of the Leap Motion field of view.

The second function, pointable, is what finger that is doing the gesture of screen-tap or key-tap. Here, like the other gestures, individual fingers can be preferred or ignored when it comes to coding the project.

The third and fourth functions include position and progress. The position vector records the location where the screen-tap or key-tap is registered. This is one of the most essential functions since the data from this vector will determine what object is being interacted with from the tap due to its location. Unlike the circle gesture, the progress function for the taps is always a value of one. There aren't any intermediate states for the screen-tap or key-tap, it's either recognized or it isn't.

4.3 Temperature Regulation Subsystem

4.3.1 Microprocessor Specifications & Function

The Arduino Uno is an open-source electronics prototyping platform. The Uno will take in the temperature and humidity sensor data in order to regulate the temperature regulation system. Next, it will be serially connected to the central PC in order to keep the PC updated on the temperature status. It's the perfect microcontroller for the mirror due to the fact that it meets all the necessary specifications required to complete its task without much overkill. An overview of the specifications is shown below in Table 4.3.1.1.

Clock Speed	16 MHz
Voltage	7-12 V
EEPROM	1 KB
SRAM	2 KB
Digital Pins	14
Analog Pins	6

Table 4.3.1.1 – Arduino Uno Specifications – Data provided from Arduino Website [41]

A standard Arduino would do the necessary for the Smart Mirror, but we wanted to take it a step further. We chose to create a "custom" version of the arduino by printing and populating our own board. This "custom" version is essentially a stripped down model that only makes use of the components needed for the smart mirror function. The main aspects of the arduino to keep in tack included the USB serial components as well as two analog pins, and the ATmega328 processor. The Eagle file that was modified is shown in Figure 4.3.1.1 below. From here, pieces were stripped off that didn't belong to the functions we needed the Arduino to perform. The way the Arduino Uno works is by being loaded with custom Arduino software, usually written in C or C++ [41]. Here the code can be used to interact with the various pins and write different values, representing voltage, to them. The code will wait for the central PC to send a serial data message with instructions on whether the user wants to adjust the lighting to a different brightness setting. At any other time, the program will simply wait for instructions while running its other functions.

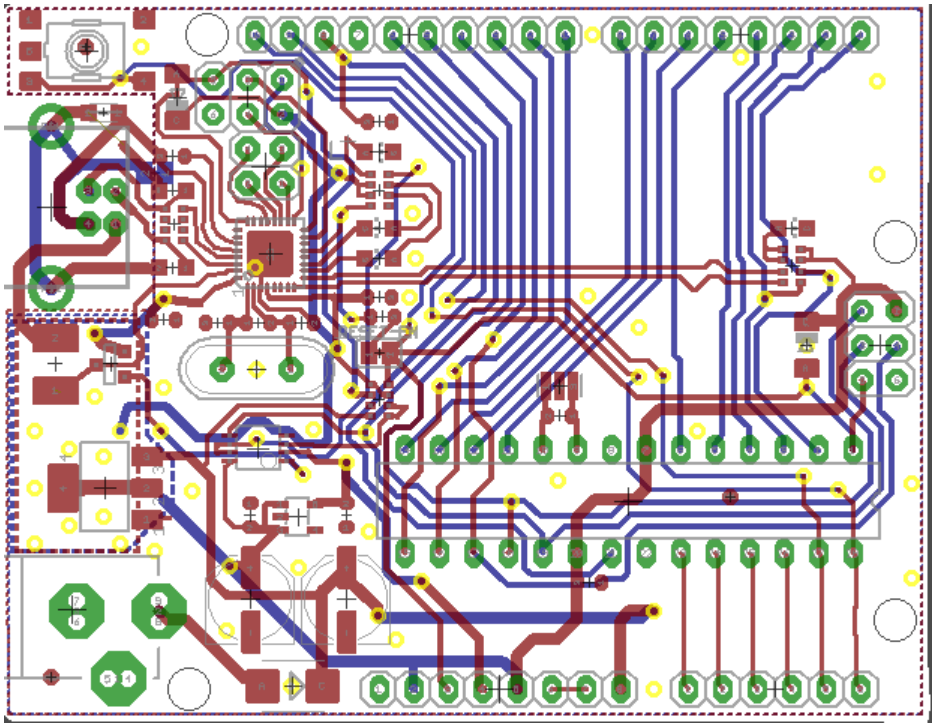

Figure 4.3.1.1 – Arduino Uno Eagle Layout

The primary function of the custom Arduino is dealing with the analog data gathered through the temperature and humidity sensors. These will develop a level of voltage based on the moisture in the air and general air temperature. The level of voltage was able to be processed through the analog-to-digital converter to retrieve a digital value for their data. This data was run through the code running on the Uno to make decisions for the temperature regulation system. The arduino code handled converting the sensor data into the correct units as well as averaging the input values in order to eliminate any sampling mistakes. The temperature and humidity data was sent via the USB serial connection to the central PC where it was analyzed by the Windows 8 code. Information is requested by the PC every 10 seconds so that it is up to date and accurate. Here, a warning was able to be displayed to the user to show that the temperature was exceeding safe limits. When the temperature and humidity reached a dangerous level, the PC then executed a shutdown command to protect the hardware from damage.

4.3.2 Temperature and Humidity Sensors

As discussed in section 3.3.6 Temperature Control, the functions of advanced temperature control systems are delicate and complicated. In our circuit design, we used one a variety of resistors coupled with the TI LM35 temperature sensor and the P-14 humidity sensor. Temperature sensor sends signal to Schmitt trigger part of the circuit. This provides power to the sensor. In order to pick an appropriate temperature sensor many important specifications need to be considered such as cost, power required response time, long-term stability, and many others. The Table 4.3.2.1 below compares two common temperature sensors:

	LM35	**TMP36**
Accuracy	0.5C	2C
Cost	$1.23	$1.77
Temperature range	-55C ~ 150C	-40C ~ 125C
Nonlinearity	+/-0.18	+/-0.5
Impedance output	0.1 Ohm	-
Sensor gain	+10	-
Load regulation	+/-0.5	6
Line Regulation	+/-0.02	-
Supply Voltage	-0.2 ~ 35 V	2.7 – 5.5V
Interchangeability	high	medium
Response time	fast	medium
Long-term stability	0.08	0.4
Self-heating	low	low

Table 4.3.2.1 – Temperature Sensor Comparison – Data provided from TI LM35 Document [68] and TMP36 Datasheet [69]

Figure 4.3.2.1 below describes thermal response in still air for LM35 temperature sensor. This describes how percent of final value changes over time. It can be seen from the graph the sensor doesn't produce a steady output until 3 minutes after activation.

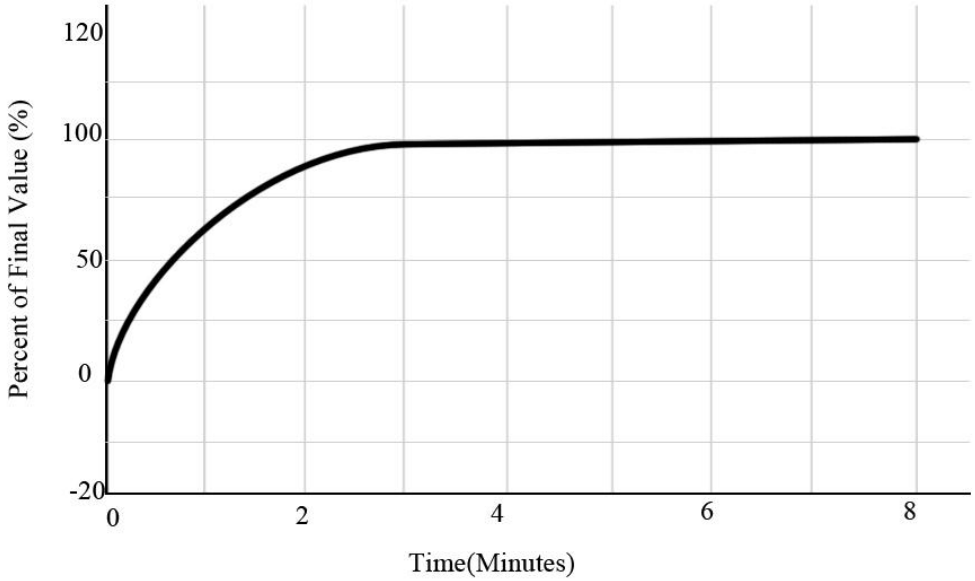

Figure 4.3.2.1 – Thermal Response in Still Air for the LM35 Sensor – Recreated with data provided from Texas Instruments [68]

Figure 4.3.2.2 below describes thermal response in still air for TM356 temperature sensor. Similarly, the graph demonstrates how the size of PCB affects the thermal response. The TM356 sensor can be seen to level out slower than the LM35 temperature sensor.

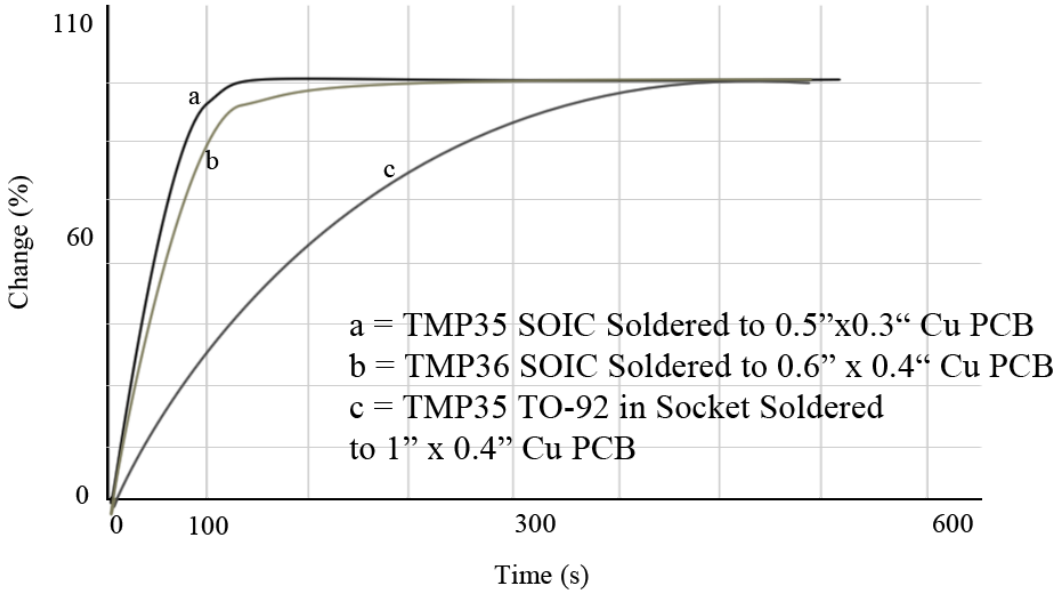

Figure 4.3.2.2 – Thermal Response in Still Air for the TM356 Sensor – Recreated with data provided from Analog Devices Data Sheet [70]

As can be seen from these two pictures, LM356 temperature sensor has a faster thermal response time. Now the temperature error needs to be compared. The temperature error for the LM35 can be found in Figure 4.3.2.3 below and the temperature error the TMP36 can be found in Figure 4.3.2.4. The temperature errors are relatively small for both LM35 and TMP36 although TMP36 is a little bit more accurate.

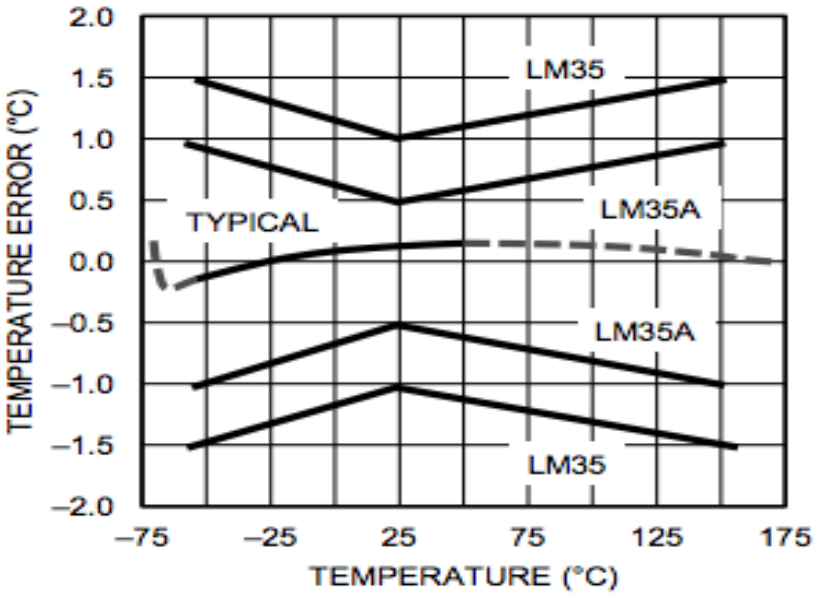

Figure 4.3.2.3 – Temperature Error for LM35 Temperature Sensor – Reprinted with permission from Texas Instruments [68]

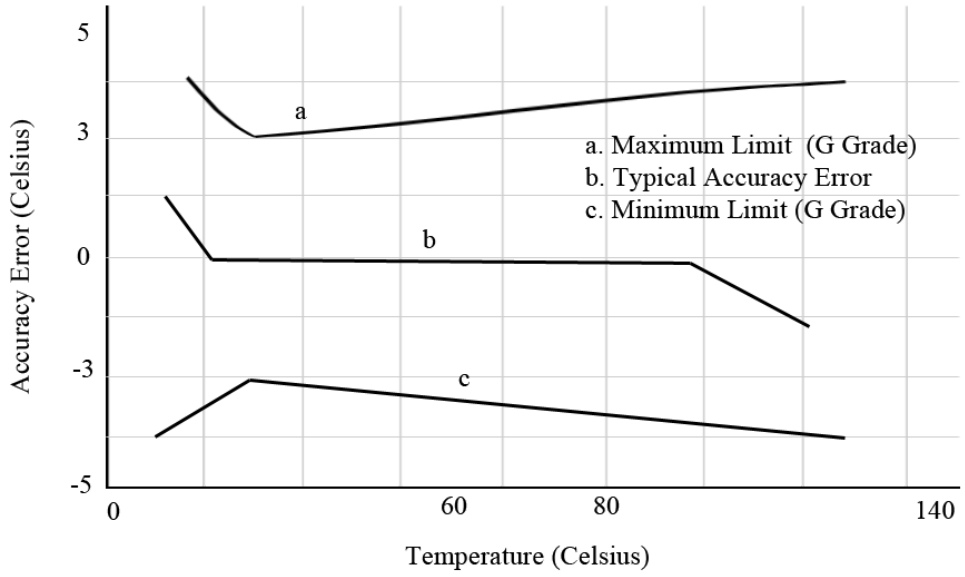

Figure 4.3.2.4 – Temperature Error for TMP36 Temperature Sensor - Recreated with data provided from Analog Devices Data Sheet [70]

In conclusion, we decided to use LM35 because it has a faster response time, it is more linear, and has a higher interchangeability.

The other sensor on the PCB is the humidity sensor. Humidity sensor sends signal to Schmitt trigger part of the circuit. The circuit, both temperature and humidity, were powered by the 12V positive input from the central PC power supply. Figure 4.3.2.5 below shows the temperature and humidity circuit from multi-sim. Following Figure 4.3.2.5 is Figure 4.3.2.6 which is the temperature and humidity circuit represented in the PCB layout format from Eagle software.

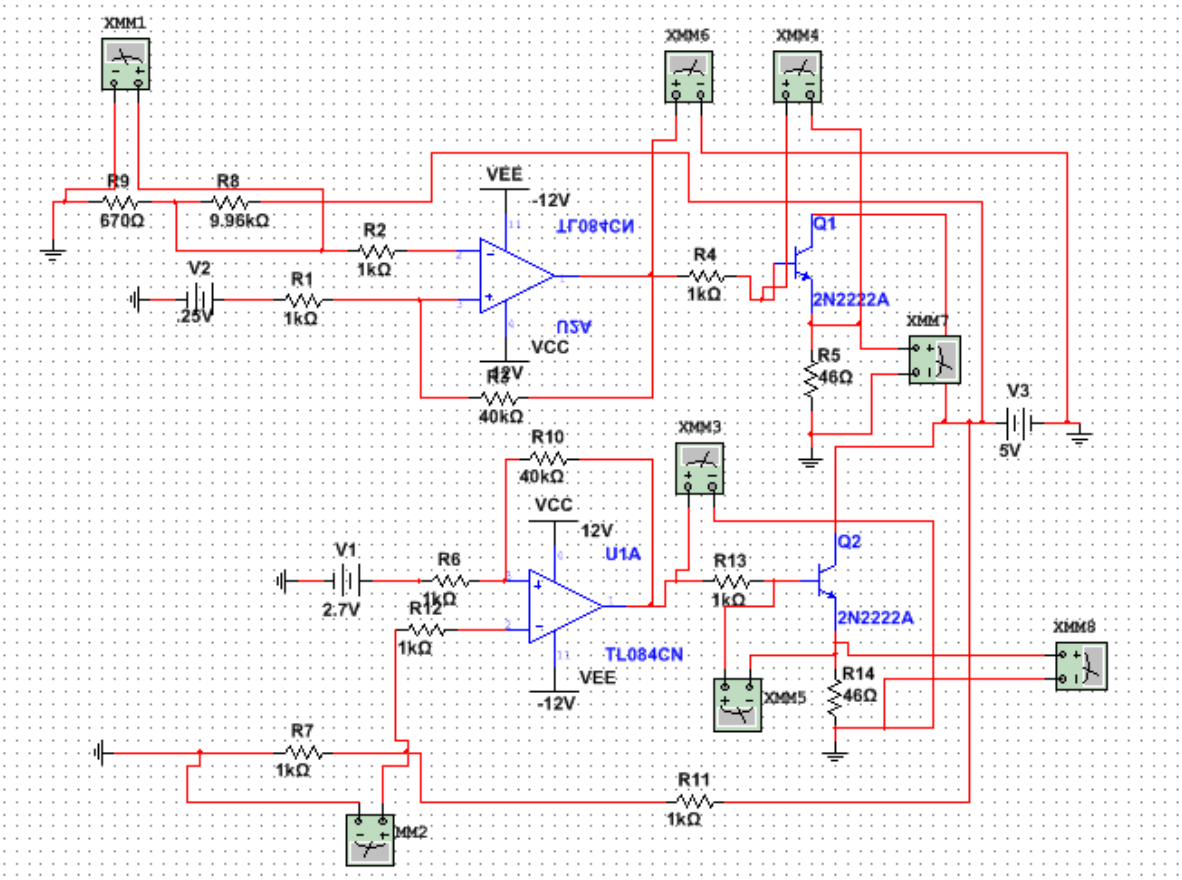

Figure 4.3.2.5 – Temperature Control PCB Circuit

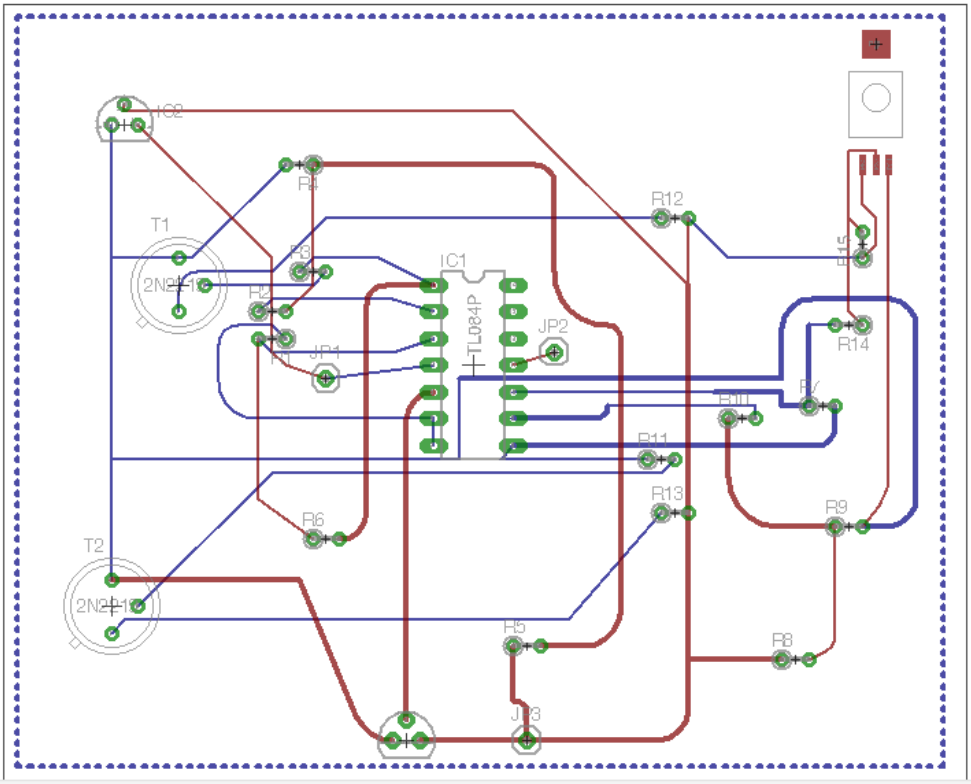

Figure 4.3.2.6 – Temperature Control PCB Eagle Layout

4.3.3 Temperature Restrictions

An important issue that needs to be addressed when designing temperature control circuit is the critical temperature of computer components. If the temperature of the system heats up to such an extent that it prevents one of the components of the system to function properly, it will damage the system. As a result, the critical temperature of every component needs to be considered so that it is protected and knows when to execute a shutdown command for the system. Table 4.3.3.1 below provides critical temperatures of system components.

Part Name	**Max Operating Temperature**
Antec 380 W PSU	50 °C
Adata 64 GB SSD	70 °C
ASRock Intel Mini ITX Motherboard	70 °C
G. Skill 4GB DDR3 1600 RAM	85 °C
Intel Core i3-4340 3.6 GHz CPU	65-69 °C
TP-Link IEEE 802.11 a/b/g/n	40 °C

Table 4.3.3.1 – Max Operating Temperatures – Data provided from Adata[64], Antec[62], ASRock[61], G.Skill[59], Intel[60], and TP-Link[63]

It can be seen from Table 4.3.3.1 that TP-Link IEEE 802.11 a/b/g/n has the lowest maximum operating temperature amongst all the parts. Therefore, the PC will turn on if the temperature of the system reaches 40 °C.

4.4 Webcam Subsystem

4.4.1 Webcam Specifications

The webcam selected for the smart mirror project is the Logitech C920 HD Pro. This webcam contains a Carl Zeiss lens that allows for a high resolution and sharp autofocus [37]. Looking at Table 4.4.1.1 below, this webcam is capable of high-resolution photos as well as video recording. Those specifications were one of the main reasons for selecting this webcam for this project since the motion detection works at a greater efficiency if the image is clearer and larger. Also in the table is the fact that this webcam contains on-board H.264 compression, which allows for a more manageable video stream with smaller sizes and less central PC stress since the compression is on the webcam. The final specification that was needed for this project was the fact that the webcam already has Windows 8 compatible drivers for easy system integration.

Dimensions	2.5 inches x 7.5 inches x 8.9 inches
Image Resolution	15.0 MP
Video Capture Resolution	1920 x 1080
Frame Rate	30 fps
On-Board Video Compression	H.264
Audio Input	Dual mics
Operating System (Min. Requirement)	Windows XP, 7, or 8

Table 4.4.1.1 – Logitech Webcam Specifications – Data provided from NewEgg [37]

4.4.2 Motion Detection Design

Motion detection was the primary job of the webcam in the smart mirror system. The webcam ran whether the smart mirror is “active” or “inactive”. Every 30 seconds, the webcam collected an image and compare that image to reference image taken the 30 seconds previously. Using some basic image filtering, motion was determined as true or false. Then the system was either set as “active” if there is motion or continue to check and be “inactive”. When the mirror is active, the mirror checks for motion every 30 seconds. Two minutes of no motion found will result in the system being set to “inactive”. When inactive, the mirror checks for motion every 5 seconds in order to recognize a user within 10 seconds.

Figure 4.4.2.1 below is the algorithm in the form of a state diagram that the motion detection software for the webcam will be using. Looking at the figure, as soon as the system is booted up, the webcam will begin checking the images. The current image is captured and a check is performed to verify that there is a reference image to compare to. If there isn’t an image, the first image taken will be set as the reference image and a new

“current” image will be taken by the webcam. In the Windows 8 API, a BitmapDecoder object must be set up to get pixel data from an image frame. Then calling the GetPixelDataAsync provides the image data of the pixels into the ProcessPixelArray [58]. Thus, multiple for-loops are performed to then compare pixel-to-pixel each of the two images. The pixels are compared in terms of intensity and color since depending on the format; each pixel will have its own value that is comparable. A threshold value was set up during coding that will adjust how sensitive the motion detection is. This threshold value will be compared against the numeric difference between the two pixels. If the difference is greater than the threshold, the coordinates of that pixel will be set as “movement”. If not, the pixel coordinate will be set as “no movement”. This will repeat from the “capture image” block in Figure 4.4.2.1 until all the pixels have been compared and checked. Here the total number of pixels marked as “movement” will be totaled up and compared against another threshold in order to limit false “movement” due to noise.

Assuming that the number of “movement” pixels is greater than the threshold, the motion variable will be set to true and the waiting variable set to zero. Once motion is found, the system checks whether the mirror is set to “active”. If yes, the system waits 30 seconds and then repeats the whole process from the beginning. If no, the mirror is set as active, waits 5 seconds, and then repeats the process. On the other hand, if no motion is found, the same question is asked. If the mirror is active, then the system waits 30 seconds, increments the waiting variable by one, and then checks the value of the waiting variable. Once the waiting variable is 4 or greater, that means at least 2 minutes have passed since motion was last found so it is safe to say the user is no longer present. This then prompts the system to go into “not active” mode. Otherwise, in either choice, the webcam will continue to check for motion. Finally, once the mirror is inactive and no motion is being found, the default process is to wait 5 seconds then repeat the whole algorithm again.

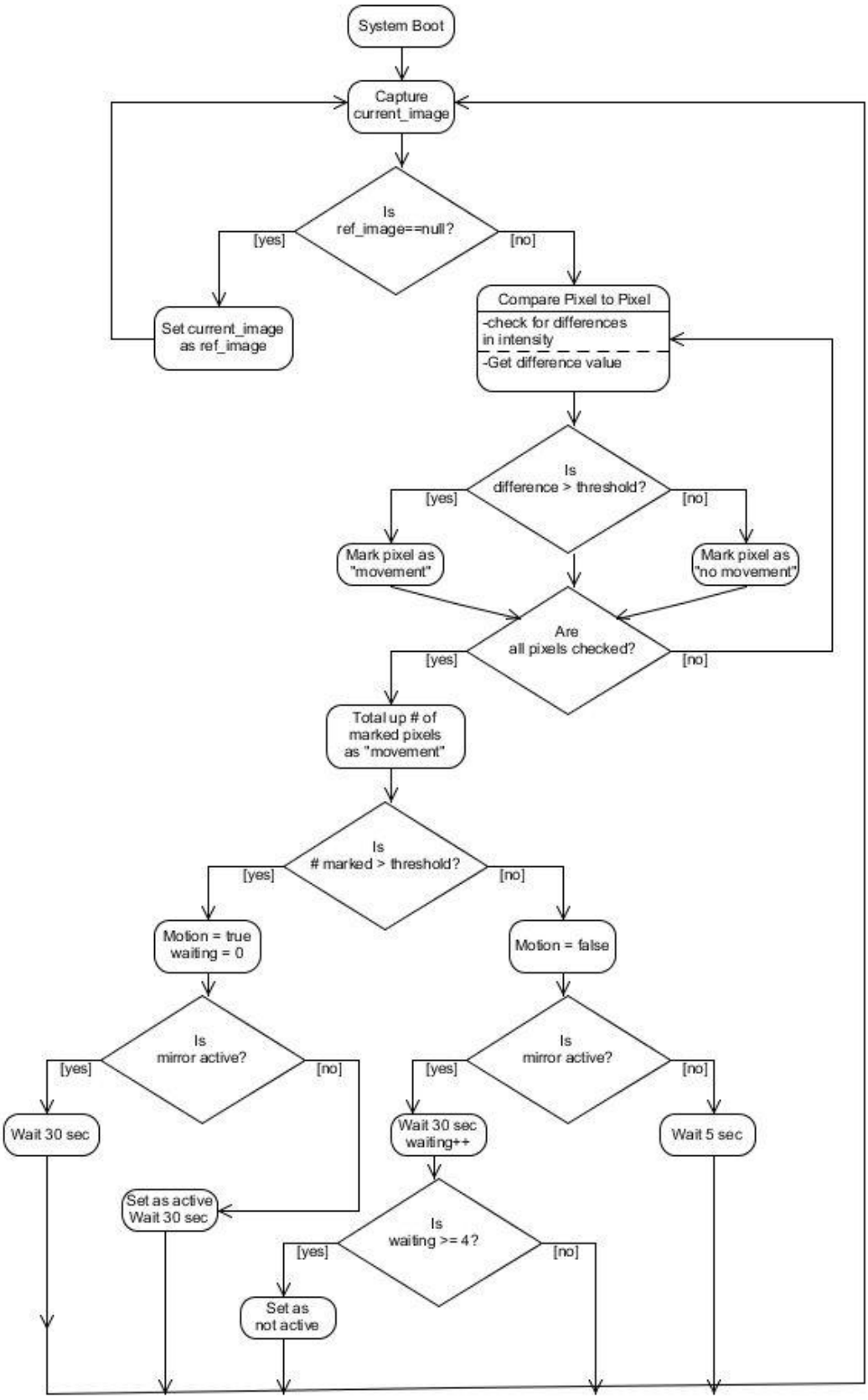

Figure 4.4.2.1 – Motion Detection State Diagram

4.4.3 Voice Recognition

Voice recognition was the second job of the webcam since it makes use of the built-in microphone. The microphone has a dual-mic setup for better voice recording. The voice recognition state diagram can be seen below in Figure 4.4.2.2. By default, the smart mirror isn't listening for any voice commands.

When the user performs the correct gesture, the voice controller leaves the default state and moves to record 5 seconds of audio from the user. This audio is saved to a WAV format and uploaded to the AT&T Speech server. The server translates the voice to text and returns what it hypothesizes the user said. If the returned text is empty or doesn't contain any of the command key words, then the program exits back to the default voice state. When a correct command is said, the program moves on to determine what keyword was stated. The key words include "tweet", "task", "play", "pause", "next", "previous", and "stop". The keyword "tweet" involves posting a tweet to Twitter. The keyword "task" involves adding a new item to the user's to-do list. The other keywords are all commands to control the music. So, if the user uses a Twitter or to-do list keyword, they are next prompted to say whatever tweet or task they wish to add. Once this is converted to text, the text is displayed to the user in order for them to either confirm with a "yes" or deny with a "no" if the text translated correctly. If it isn't correct, the action is canceled and the system returns to the default voice state. If it is correct, the action is performed. Regarding the music commands, they are processed similarly except there is no confirmation. The keyword located in the voice command text is executed directly such as "play music". This statement will instantly begin playing an music located on the smart mirror hard-drive.

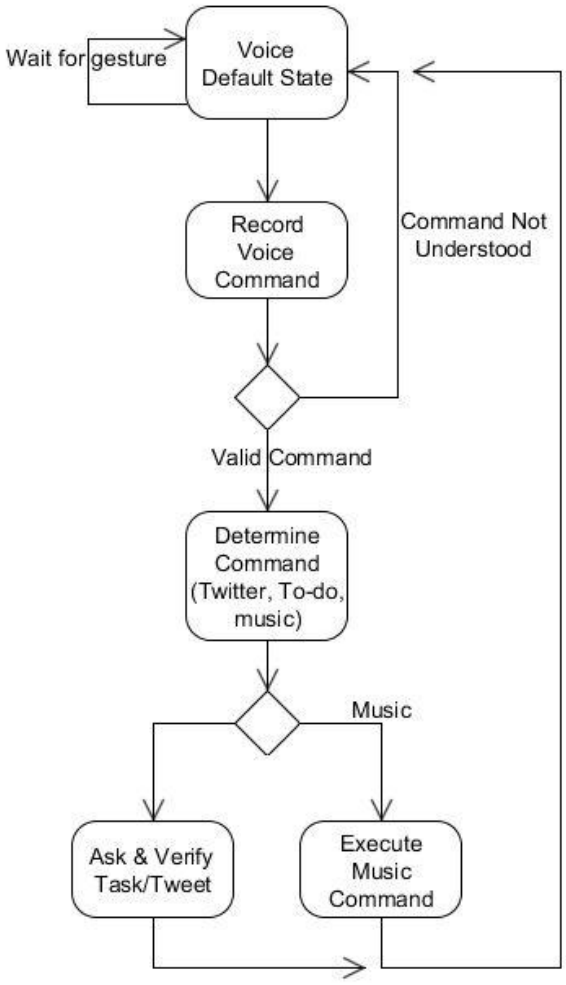

Figure 4.4.2.2 – Voice Recognition State Diagram

4.5 Control Software Subsystem

4.5.1 Applications

The design of the control software for the user focused on six applications. These applications include weather, music, twitter, to-do list, calendar, and news. They will each be interactive by gestures from the user. This section will outline the basic design and code flow for each application. For now, background tasks such as the gesture control and voice control will be ignored in terms of how they are handled within the code.

The weather application provided local weather data to the user. The GUI in normal mode will display the weather location, the current temperature and today's high and low values. When in expanded mode, the day's hourly forecast will be displayed along with the forecast for the next two days. This application is simply pulling data from the

National Weather Service server into the application and saving it into the local variables used for the GUI. The Windows 8 SyndicationClient was set up to query the RSS URL for the weather in the particular location [58]. The XML from the weather service is then received from the async task call. The XML is parsed for all the necessary information which is stored into variables. These variables are then synced with XAML which updates the GUI for the user to see the weather data updated.

The music application allows the user to play music out of the speakers while using the smart mirror. This music played is stored locally on the PC hard-drive so the user must provide their songs. The actual music app in normal mode displays the title of the song and album art if available. In expanded mode, a list of all the songs currently located in the user's music folder is displayed. The controls for the music app are all done by use of the user's voice. Voice control includes the ability to vocally dictate the music to stop, play, pause, previous, or next. In terms of the application code, first the app will query the local storage for music by getting the files async. These music files will be appended to a playlist for easy access. The songs data such as title and artist are saved as well as their album art. The player starts at the beginning of the playlist when the user begins playback. The song index is either incremented or decremented if the user choses to play the previous or next song in the playlist. Using the MediaStreamSource and XAML Mediaplayer object allow for the music to play in the background of the app, no matter what "menu" the user is at [58]. Once playback is executed, the GUI is updated with the song information and album art.

The Twitter application provides the latest tweets from the user's timeline and also allows the user to post a tweet using voice recognition. In normal mode, the Twitter app displays the most recent tweet from the home timeline. In expanded mode, the app displays the 3 most recent tweets. The app communicates with Twitter's servers by use of the Twitter Rest API [57]. This describes the steps that need to be taken to get authorization to their servers and setup communication. A HTTP Get request is used to get the request token, and then the user is prompted to sign into their account within the smart mirror through use of the WebAuthenticationBroker namespace. Next, the returned data is sent back to get an access token which is used for any further communication with the Twitter server. This follows the OAuth 1.0 framework. Another HTTP Get request is used to get the latest tweets from the timeline and a HTTP Post request allows the user to post a tweet to their home timeline. This data is returned and saved to be sent to the XAML to be displayed to the user.

The to-do list application displays to-do list items, or tasks, to the user. In normal mode, the to-do list app displays 5 items from the user's to-do list. In expanded mode, the app will display up to 10 items. The app is synced with a popular to-do list application called Toodledo. This app is a popular productivity application that is available on iOS, Android, and BlackBerry devices [40]. There is even a website interface. The developers provide a free, open source API that allows for integration and syncing of their app into various platforms. As mentioned earlier, theToodledo uses OAuth 2.0 framework for their server communication. The user signs in using the WebAuthenticationBroker namespace and an authorization token is returned. Then through use of the authorization

token, or a refresh token, an access token is requested. This is returned with a timestamp within a JSON format. This access token is then passed in further requests to get new tasks and add new tasks. The tasks are returned in a JSON format which is parsed and the values needed are saved locally. The code automatically updates the tokens as needed in order to prevent unneeded re-logins from the user. The saved data is passed to the variables responsible for updating the XAML GUI.

The calendar app provided the user's events for the current day and next 2 days. The app use the user's "iCal" file which is a text file ending in ".ics" [56]. Such files are obtaining from calendars such as Google, Apple, and others. The URL link to the file is provided by the user to the smart mirror. The smart mirror uses async tasks to download the file from the link in the background. Once the file is downloaded, it is opened and parsed for specific data. The custom parser checks the date information and extracts the event's time and description. This data is appended to local vector variables for organization and is later used by the XAML variables to display the data to the GUI.

The news application provides the latest news stories from Bing News. In normal mode, the application displays the most recent new story title and story image. In expanded mode, the news story description along with the title and image are displayed. This app will essentially be an RSS reader which pulls in the RSS data provided by Bing. The Windows 8 SyndicationClient was set up to query the RSS URL for the weather in the particular location [58]. The RSS is parsed for the story data as well as the link to download the news image in the background. Finally, the data is moved to the XAML for display to the user through use of local variables.

4.5.2 Foreground and Background Task Processing

The central PC is the "heart" of the system. It is the main point where most information is relayed to and from. As can be seen from the system state diagram in section 4.1 previously, the Leap Motion controller, webcam, MCU, TV, and speakers are directly interacted with by the central PC. The processing of all these data will be divided into two task processing sections. These sections are the foreground and background processing pools. The foreground task involves managing the GUI elements with the logic elements of the background tasks. The primary interactions will involve the GUI with the gesture recognition processing. The state diagram of how the GUI reacts and works with the Leap Motion gestures can be seen in Figure 4.5.2.1 below. Here the system first boots up and the first item to display on the screen is the time. Next, the apps open to normal mode, the default boot up state. From here, depending on whether the gesture is a swipe or tap will determine what the GUI does next. The correct swipe on the left-side of the screen will change those apps into the minimized state, leaving the right side in normal mode. The appropriate swipe on the right side will change those apps into a minimized state. The states are circular so the user can make either one side or the other normal to return to having all the apps in normal mode. When all are in normal mode, a long point and hold on the app will put it into expanded mode. A quick vertical swipe up on the app will return it to normal mode.

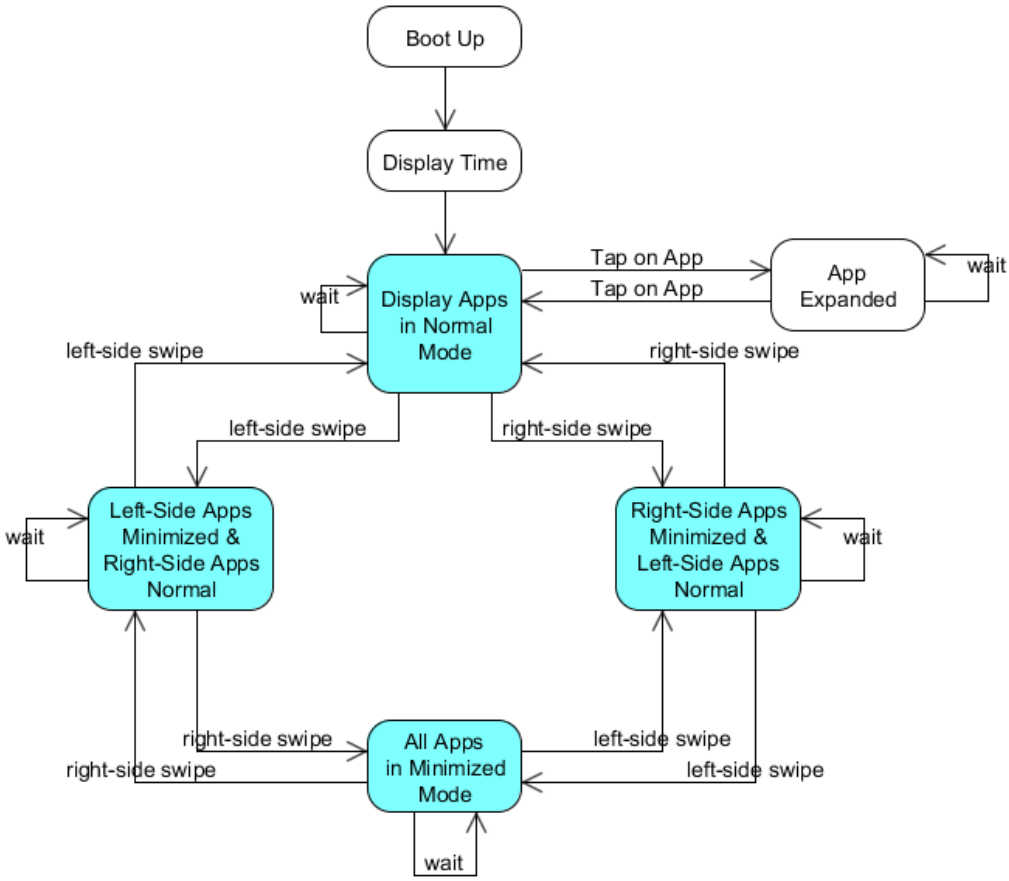

Figure 4.5.2.1 – Foreground GUI State Diagram

When it comes to the background task processing, there is a lot of data being manipulated almost constantly. Peripherals such as the Leap Motion controller are constantly updating to check if there is gesture being performed within its field of view. The webcam is also frequently taking images to compare in order to determine motion. To reduce unnecessary data processing, the webcam only searches for user motion every 30 seconds. These interactions have to be appropriately timed and a sample sequence diagram for these interactions on the central PC is shown below in Figure 4.5.2.2. The 6 essential entities tied to the central PC are listed with block instructions shown of what they should and will be doing most of the time. On boot up, the central PC will start by loading the GUI and in turn will send that data to the TV screen behind the mirror. Once that is done, the 4 asynchronous tasks will start. Each task will be added into the asynchronous thread pool provided by the Windows 8 API. The asynchronous aspect will allow the user to continue using the mirror without the GUI freezing up. The multi-thread aspect suggested allows for a closer to parallel execute of each of the code than a linear programming technique would allow. This parallel-type style allows for the Leap Motion and other parts to be handled as soon as data is delivered. The sequence diagram shows the interactions over a period of time and the tasks each part will be performing. All the

data is relayed to the central PC which then processes it through the algorithms and code to then either update the GUI or update another peripheral.

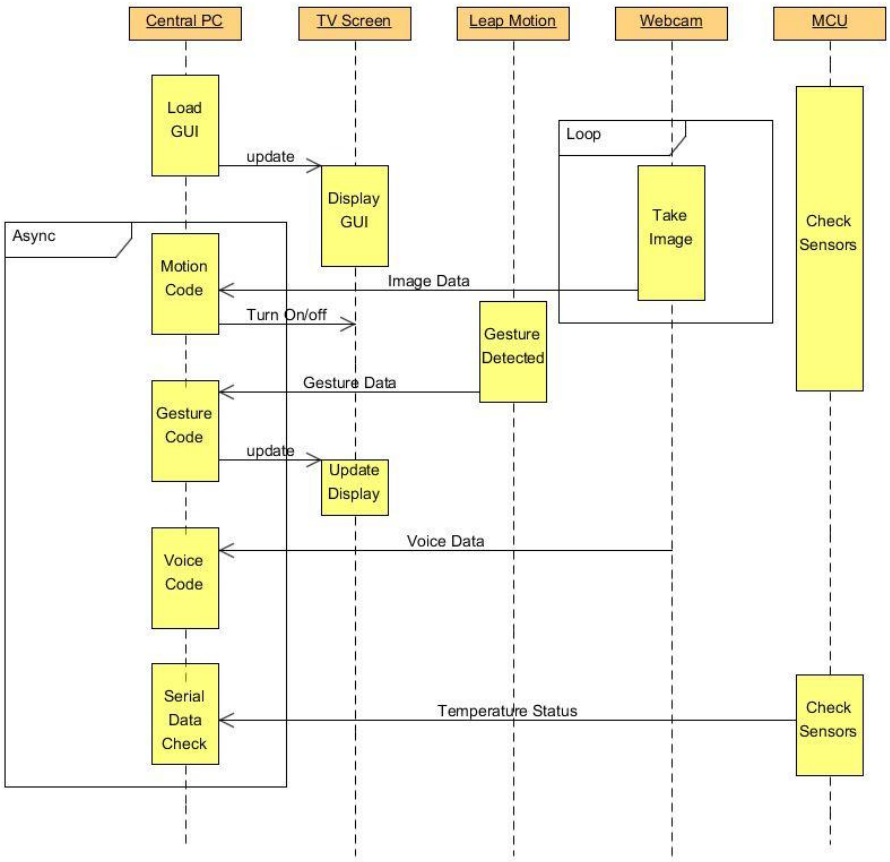

Figure 4.5.2.2 – Central PC Sequence Diagram

While the sequence diagram shows the interaction across all the peripherals, the state diagram for the Central PC in Figure 4.5.2.3 below shows specifically what is going on within the Windows 8 application running on the PC. The diagram demonstrates how the application runs from boot-up to shut-down. On boot-up, note this isn't from a non-active state, the mirror is set to active and the program checks for user data for the apps that need it. These apps include the Twitter, calendar, and to-do list app which all require an account setup externally from the mirror. Once those accounts are established, the app data is loaded. From here, the asynchronous thread pool is started which adds all the appropriate functions into the Windows thread pool which essentially optimizes the code on the processor. The first task added is the GUI processing which displays the GUI to the user. This contains two states which change based on whether the mirror is active or not. As stated earlier, the asynchronous behavior of the Windows 8 architecture and threading allows for the GUI to unaffected by logic processing in the background. This is done by having its own thread to run on, separate from the other processes. Next, the gesture update function is added which checks for gestures in the field of view and executes commands that alter the code and GUI for the user. The image processing function is added to the pool after which starts to check for user motion to keep the

mirror active or begin the process of changing it to inactive. The diagram shows how the image processing is halted for 30 seconds in between images to prevent unnecessary image processing stress on the system. After, the serial data check function is added to check the status of the temperature system that is being relayed from the MCU. Nothing is done if the status is normal but if the temperature becomes unsafe, a warning to the user is displayed stating that the system is too hot and needs to shut down. Once that message is displayed, the system proceeds to shut-down. Also in the thread pool is the code that deals with the voice commands. Here the user activates the mirror to start listening and they state a command. If the command is valid, such as "play music", it is executed. Finally, if music is started, this process is sent to the background so that it can continue playing while the user accesses other parts of the mirror GUI. This is the only task that is removed from the thread pool once the stop command is run.

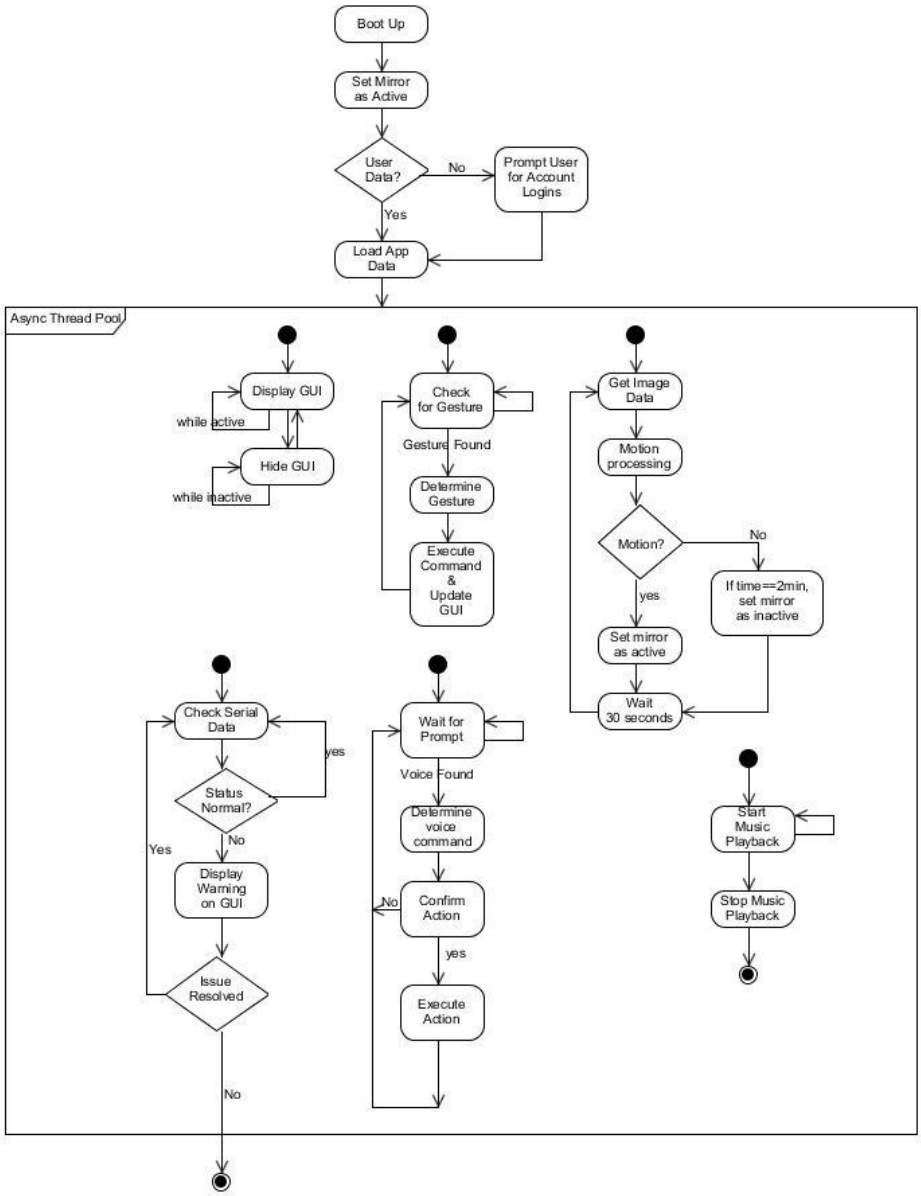

Figure 4.5.2.3 – Central PC State Diagram

4.5.3 User Interface and GUI Design

The user interface and GUI design are specifically designed for the user. The GUI should visually make sense, be unobtrusive, and be easy to read. The user interface that compliments the GUI should make sense, provide options to the user, and be intuitive. These are the elements that make or break a product in the consumer world. The smart mirror project is intended for consumer use so it must be all those standards in terms of both functional and non-functional design.

As stated earlier, there are 6 applications plus the clock. These must be visibly available to the user and capable to adapt to what the user wants to do and see. Referencing Figure 4.5.3.1 below, this would be the default layout of the applications. The time will be top and center right under the webcam. Here the time will be able to be constantly displayed to the user without covering up any mirror space that would normally be of use. A modern, flat font will be used to be up-to-date on current design trends and stay in theme of all other Windows 8 applications. Sample application icons can be viewed below in Figure 4.5.3.2 which contains 6 app icons that are first renditions of what will be used in the final smart mirror GUI. This flat, vector design with bold colors will translate to the application icons as well that appear on the left and right edges of the mirror. There will be 3 applications on each side, in a vertical layout. When the applications are displayed as only an icon, they are in the minimized state. There are three states: minimized, normal, and expanded. Minimized will show no information, normal will show a small amount of information, and expanded will show the most information to the user.

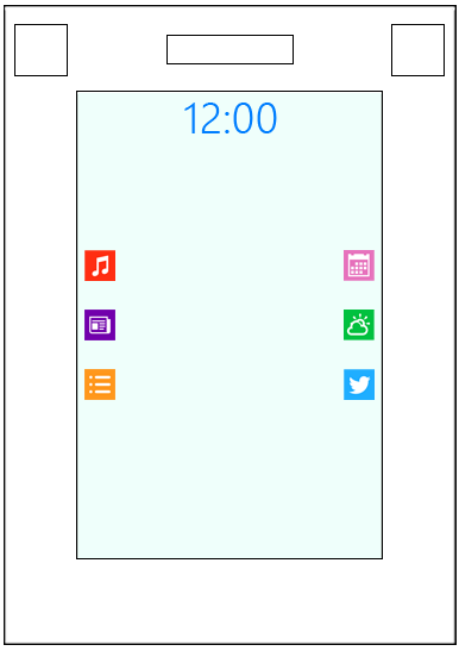

Figure 4.5.3.1 – Smart Mirror with Apps Minimized (Not to Scale)

Figure 4.5.3.2 – Sample Flat, Bold Color Icons – Vector Icons provided by FlatIcon.com

Now looking at Figure 4.5.3.3 below, each application has been expanded into its normal state. The normal state contains the identification of the application through icon display and then displays the content expected of the application. Each app will feature 2 to 3 lines of data such weather, news articles, or tweets. That data replaces the "Text Here" labels in Figure 4.5.3.3. This mode is geared to be informative but unobtrusive from the main usable area of the mirror. The design of the user interface is made so that the user can easily switch between minimized and normal mode frequently based on their needs.

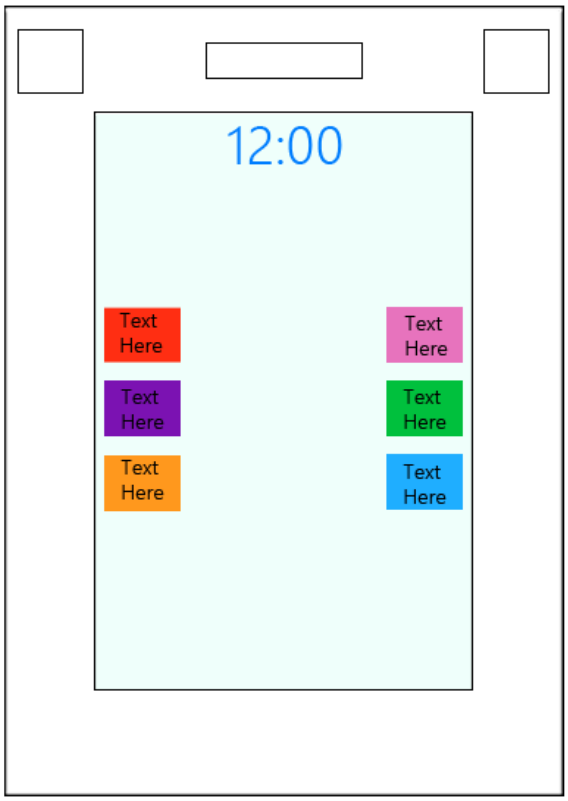

Figure 4.5.3.3 – Smart Mirror with Apps in Normal Mode (Not to Scale)

Looking at Figure 4.5.3.4 below, this shows an example of an application in expanded mode. This mode can only be accessed by manually selecting a particular app through a "point and hold" gesture on the app in normal mode. The expanded mode takes up more mirror "screen" space and is intended for use when the user has a direct interest in being

provided more information on a particular application. Only one app can be in expanded mode at a time because when in expanded mode the application will animate to the center to the screen for direct viewing by the user. In expanded mode, more data is provided than in normal mode. For example, multi-day calendar events are displayed for the calendar app or a multi-day weather forecast for the weather app. Each application's expanded mode can be reduced back to normal mode with a quick vertical swipe gesture.

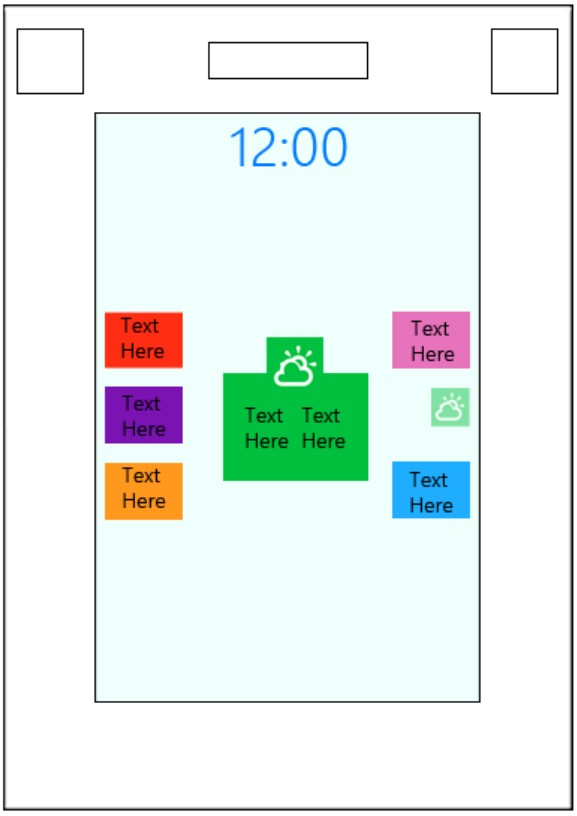

Figure 4.5.3.4 – Smart Mirror with an App in Expanded Mode (Not to Scale)

4.5.4 User Interface Gesture and Voice Controls

The user interface can be organized and designed well but how the user interacts with it is the real key to a successful, intuitive, easy application. The user uses specific, custom gestures read in by the Leap Motion controller. The code takes advantage of the data relayed in each frame by the SDK and truly makes it a painless and touch-free experience. The full list of gestures that can be performed and their UI action can be found in Table 4.5.4.1 below.

Gesture	Action
Left Swipe (Left of screen center)	This changes all 3 apps on the left-side of the screen from normal mode to minimized mode. This is performed with one hand.
Right Swipe (Left of screen center)	This changes all 3 apps on the left-side of the screen from minimized mode to normal mode. This is performed with one hand.
Left Swipe (Right of screen center)	This changes all 3 apps on the right-side of the screen from minimized mode to normal mode. This is performed with one hand.
Right Swipe (Right of screen center)	This changes all 3 apps on the right-side of the screen from normal mode to minimized mode. This is performed with one hand.
Swipe Up	This closes the app currently in expanded mode. This gesture can be performed anywhere with one hand.
Point and Hold	This allows for individual interaction between the individual apps. Use of this gesture on an application in normal mode will change it to expanded mode.
Clockwise Circle	This forces a manual data update, or refresh, for any application using data from a third-party web server. For example, the weather app.
Counter-Clockwise Circle	This activates the voice recognition mode, where the mirror will begin listening for a voice command from the user.

Table 4.5.4.1 – List of Leap Motion Gestures and UI Actions

The gesture controls are a combination of swipes, long points, and circles. Since the swipe gesture is the simplest to perform, they are the majority of the controls. The user is essentially swiping away and swiping back the applications between their normal and minimized state. The swipes focus on controlling what the user sees and wants to see. The point and hold is used when the user wants more. That is why the user only accesses the expanded state when the application is chosen with the point and hold. The least often needed actions use the more complicated gesture, the circle. The voice control uses the counter-clockwise circle in order to prevent unintentional activation of the voice recognition system. Finally, the application refresh, or manual update, uses the circle gesture. More often or not, the applications will update correctly and promptly enough that the gesture won't be needed. There is always a possibility that the communication with the server misbehaviors so the option to manually refresh is a good option to have to avoid user frustration.

Besides the gesture controls, there is also voice controls. There are 7 unique commands that the user can state for the smart mirror. These 7 commands are listed in Table 4.5.4.2 below. When the user does the appropriate gesture, the mirror will wait for 5 seconds for

a command to be said. Of the commands, 5 of them are related to controlling the music. So if one of those is stated, such as “play”, the music will respond appropriately unless the keyword isn’t included in the user’s statement. The other two actions are posting a tweet to Twitter and posting a task to their to-do list. Upon beginning the prompt, the user will include the keyword “tweet”, “twitter”, “task”, or “list”. If understood, the mirror prompts the user to say their tweet or task. Then the tweet or task is converted to text and displayed back to the user for them to confirm or deny that it is correct through use of the keywords “yes” or “no”. Once completed, the tweet or task is posted to the appropriate server.

Voice Command	Action
Say “Tweet” or “Twitter”	Initializes the command to post a tweet to the user’s timeline. A follow-up prompt asks for the specific text of the tweet.
Say “Task” or “List”	Initializes the command to post a task to the user’s to-do list. A follow-up prompt asks for the specific text of the task.
Say “Play”	Initializes the command to play the currently selected song. Defaults to first in playlist.
Say “Pause”	Initializes the command to pause the currently playing song.
Say “Stop”	Initializes the command to stop the currently playing song.
Say “Previous”	Initializes the command to play the previous song in the playlist.
Say “Next”	Initializes the command to play the next song in the playlist.
Say “Yes” or “No”	Used to confirm or deny that the tweet or task is correct before posting to the server.

Table 4.5.4.2 – List of Voice Commands and UI Actions

4.5.5 Microprocessor Software Overview

The software running on the custom Arduino MCU is Arduino code. The processor, the ATmega328p, has flash memory so it was programmed and attached to the smart mirror custom MCU PCB. The code is simple but important to the overall performance and functionality of the smart mirror project.

The code on setup sets the appropriate analog pins and opens a serial connection with a 9600 baud. Once completed, the program enters the main loop. Here in the main loop, first the analog temperature data is sampled from the assigned pin. In order to prevent extraneous and inaccurate data, the temperature sensor is sampled 8 times. This temperature data is then averaged and converted to Celsius. Next, the humidity is sampled multiple times as well for an accurate reading. The humidity is then put through a series of conversions that takes the ADC data and puts into the correct percentage format. This process is repeated after a short delay. The control PC checks the temperature and humidity data every 5 seconds. The control PC sends a request character serially to the MCU and the MCU gets interrupted to return the most recent temperature and humidity data back to the PC. This process is represented in a state diagram in Figure 4.5.5.1 below.

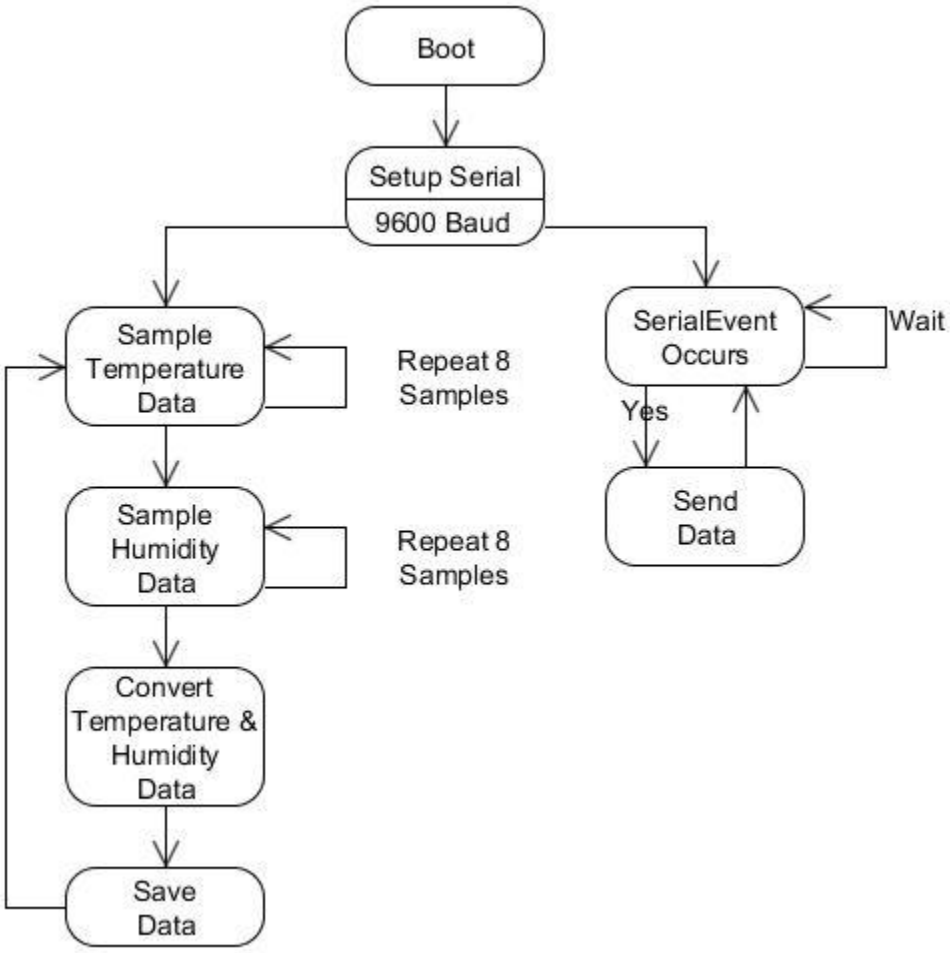

Figure 4.5.5.1 – MCU State Diagram

5.0 Design Summary of Hardware and Software

5.1 Hardware

The smart mirror design included a variety of hardware designs. These encompassed the temperature regulation circuit, the custom MCU, and their PCB layouts. In Figure 5.1.1 below is the temperature regulation circuit design. The circuit focused on two sensors, the humidity and the temperature. The humidity sensor used was the P-14 and the temperature sensor used was the TI LM35. Each was chosen for their accuracy since that was the most important aspect of their purpose. The circuit took a 12V positive input from the central PC power supply, which eliminated any need for an external power supply. This was great for keeping a smaller "footprint" as well cost saving. Through a series of resistors, voltage regulator, and diodes, the sensors were given their correct level of voltage. Once the circuit was tested on the breadboard and simulated on the computer, it was converted into its Eagle format. This Eagle layout can be seen in Figure 5.1.2 below. Advanced Circuits used the Eagle format to print the PCB and then the various components were soldered on after.

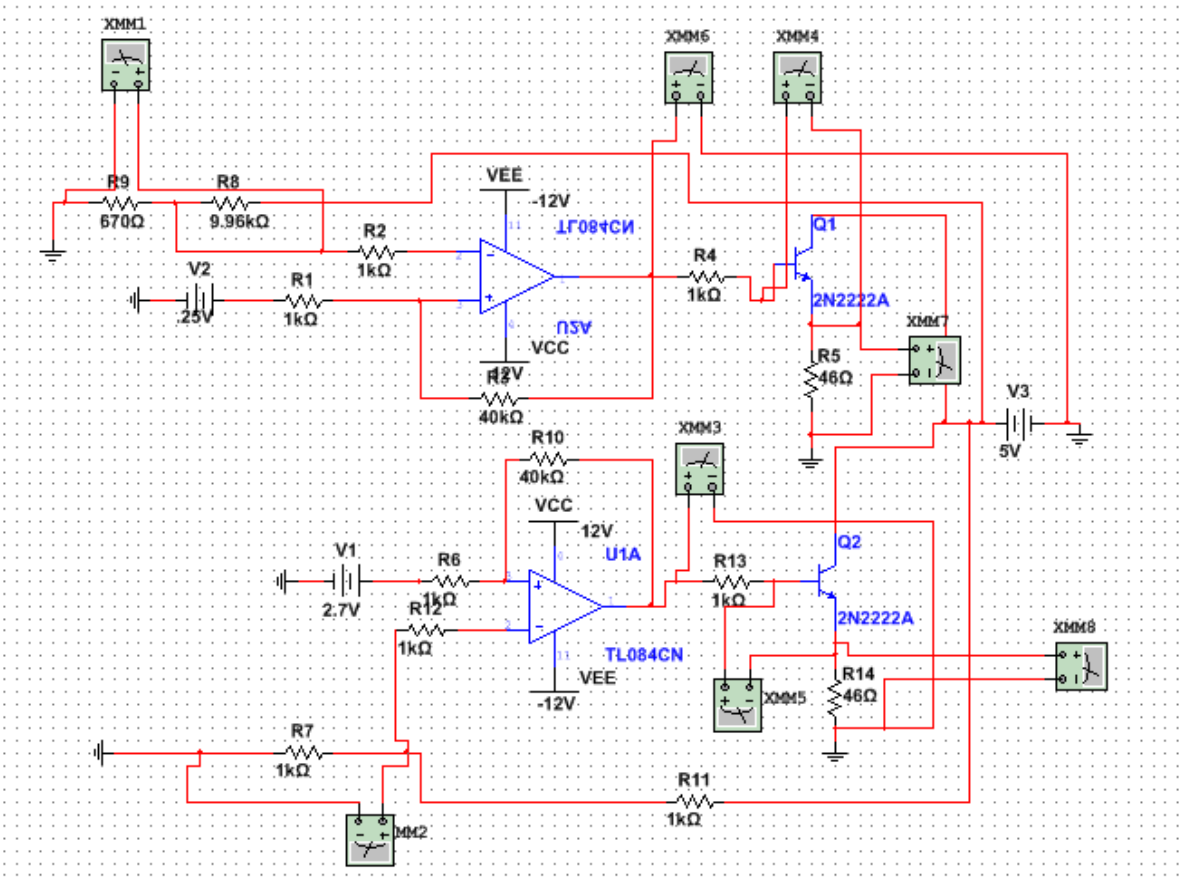

Figure 5.1.1 – Temperature Regulation Circuit Design

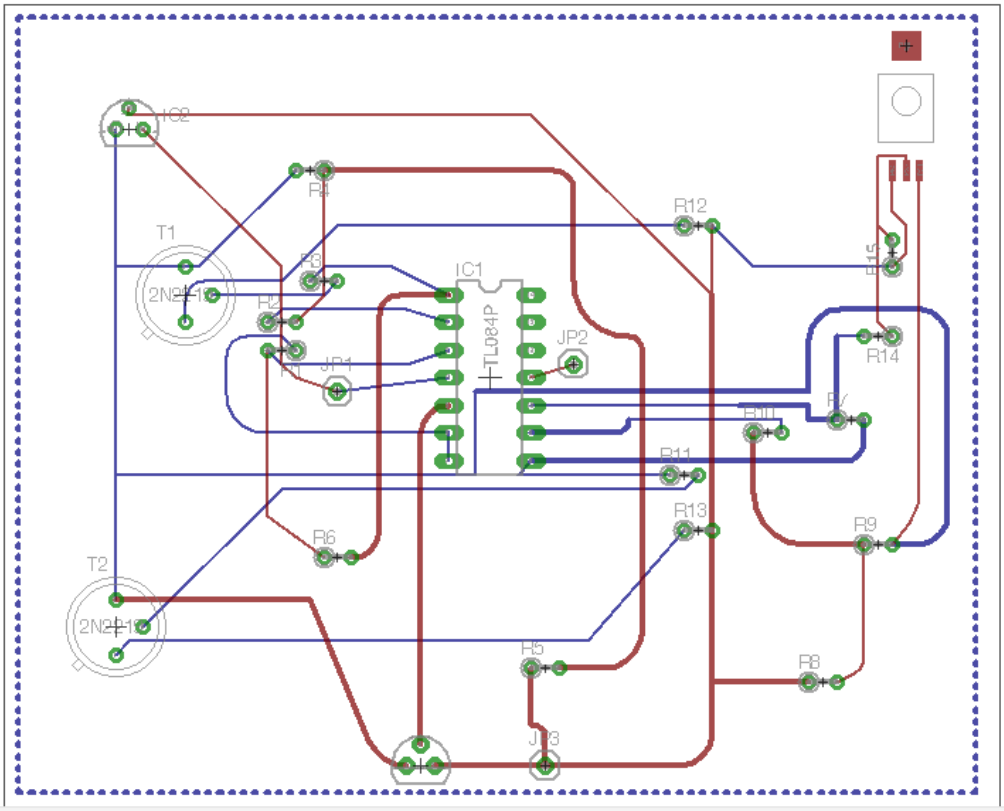

Figure 5.1.2 – Temperature Regulation Circuit Eagle Layout

Figure 5.1.3 below is the Eagle file used in creating the custom Arduino for the smart mirror. Each of these individual pieces was obtained from de-soldering them from the Arduino Uno board for both time and financial reasons. For the MCU design, the circuit was printed by Advanced Circuits using the customized Eagle file provided from the Arduino website. Once board design was printed, we soldered the components on a circuit board that we needed. The components we chose to solder were only the ones necessary for getting analog input, converting it to digital signals, and sending the data serially to the central PC by the USB. Some of the components were too small to solder ourselves and we went to Quality Manufacturing Services (QMS) for assistance.

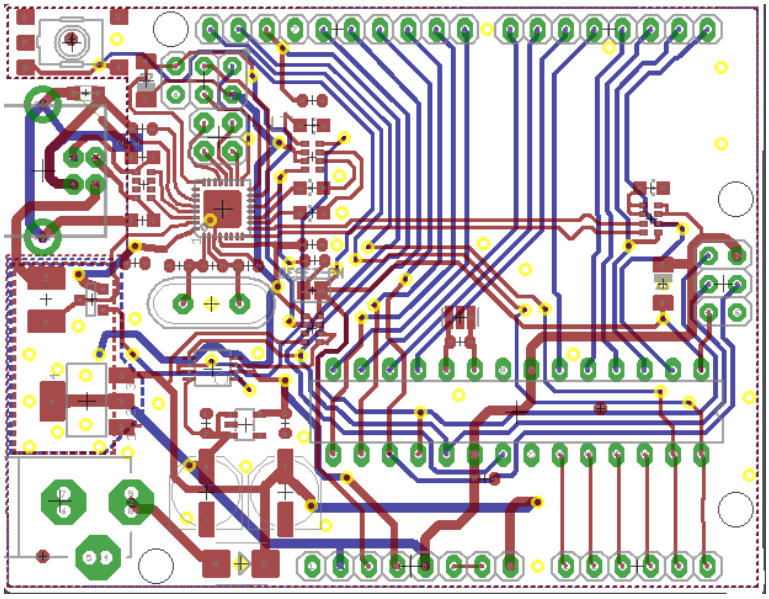

Figure 5.1.3 – MCU Eagle Layout

5.2 Software

The software block diagram gives the complete overview for all the software interactions for the smart mirror. Seen in Figure 5.2.1 below, the MCU took care of all the temperature data processing that is used in the temperature and humidity regulation system. The processing of the voice data was done via the webcam and central PC. The central PC also took care of the user gesture input, webcam image processing, and outputting the GUI to the user. The most demanding code was run on the central PC software.

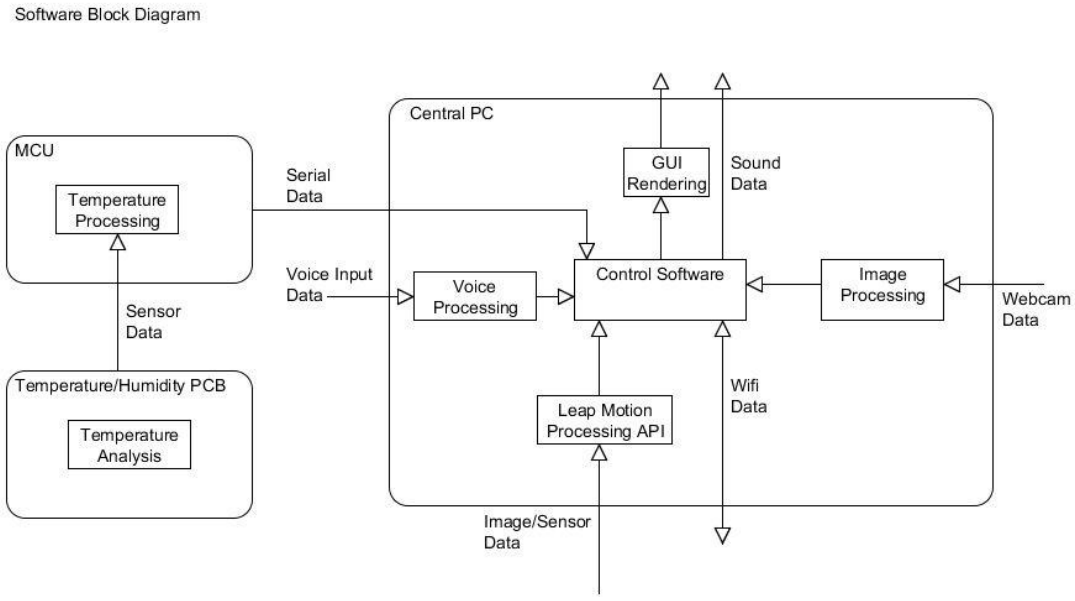

Figure 5.2.1 – Software block diagram involving MCU and PC

The primary foreground task for the central PC involved managing the GUI elements with the logic elements of the background tasks. The primary interactions involved the GUI with the gesture recognition processing. The state diagram of how the GUI reacts and works with the Leap Motion gestures can be seen in Figure 5.2.2 below. The states are circular so the user can make either one side or the other normal to return to having all the apps in normal mode. The design flow was created in way to prevent the user from being "lost" while using the smart mirror as well give them options in what information is presented on the screen.

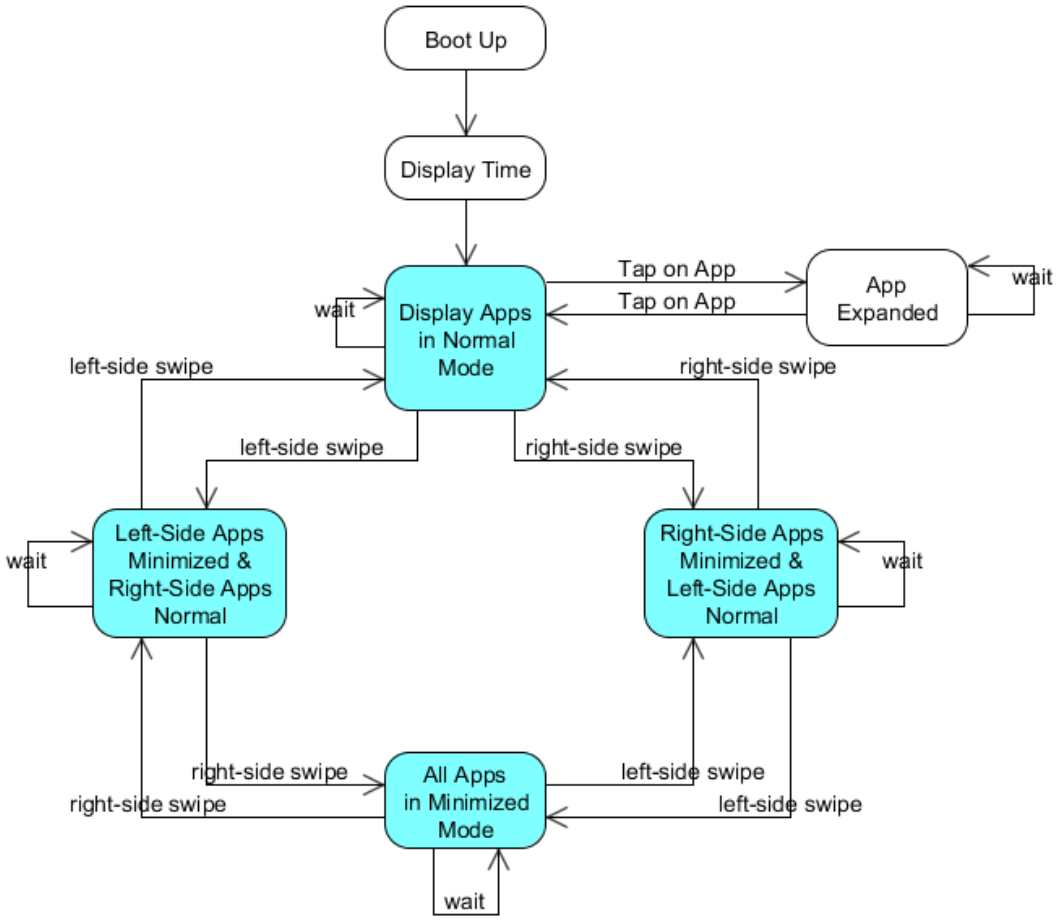

Figure 5.2.2 – Gesture Control Interaction with PC Application

For the specific GUI design, there were 6 applications plus the clock. Referencing Figure 5.2.3 below, this was the default layout of the applications. The time was top and center right under the webcam. Here the time was able to be constantly displayed to the user without covering up any mirror space that would normally be of use. A modern, flat font was used to be up-to-date on current design trends and stay in theme of all other Windows 8 applications. There were 3 applications on each side, in a vertical layout. When the applications were displayed as only an icon, they were in the minimized state. In normal mode, the applications displayed minimal information and in expanded mode,

they displayed the most information. There were three states: minimized, normal, and expanded.

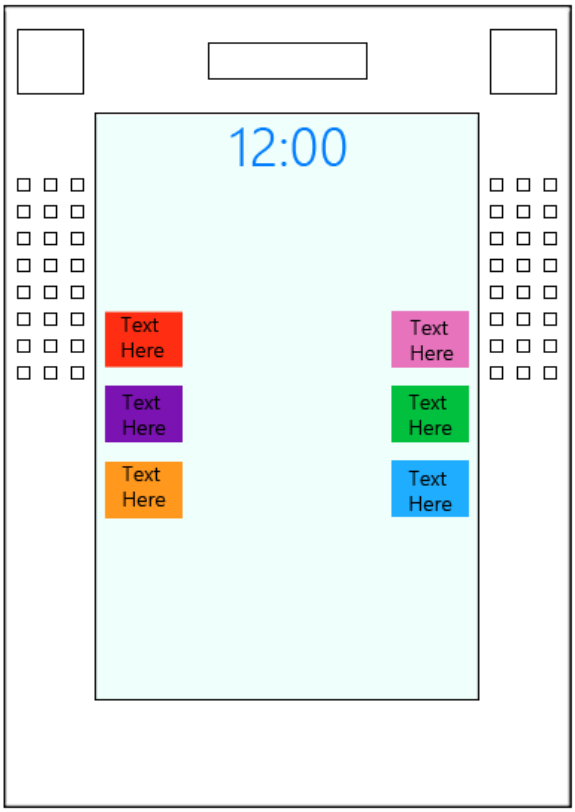

Figure 5.2.3 – Smart Mirror GUI Layout (Normal Mode)

The state diagram for the Central PC in Figure 5.2.4 below shows specifically what is going on within the Windows 8 application running on the PC. The diagram demonstrated how the application ran from boot-up to shut-down. The basics of this multi-thread algorithm involved running the update code for the components interacting with the application in parallel. This prevented the UI from being locked up to the user and allowed all the code to work almost simultaneously or asynchronously. The thread included the Leap Motion gesture code, the motion detection code, voice processing, MCU interaction code, and music playback code.

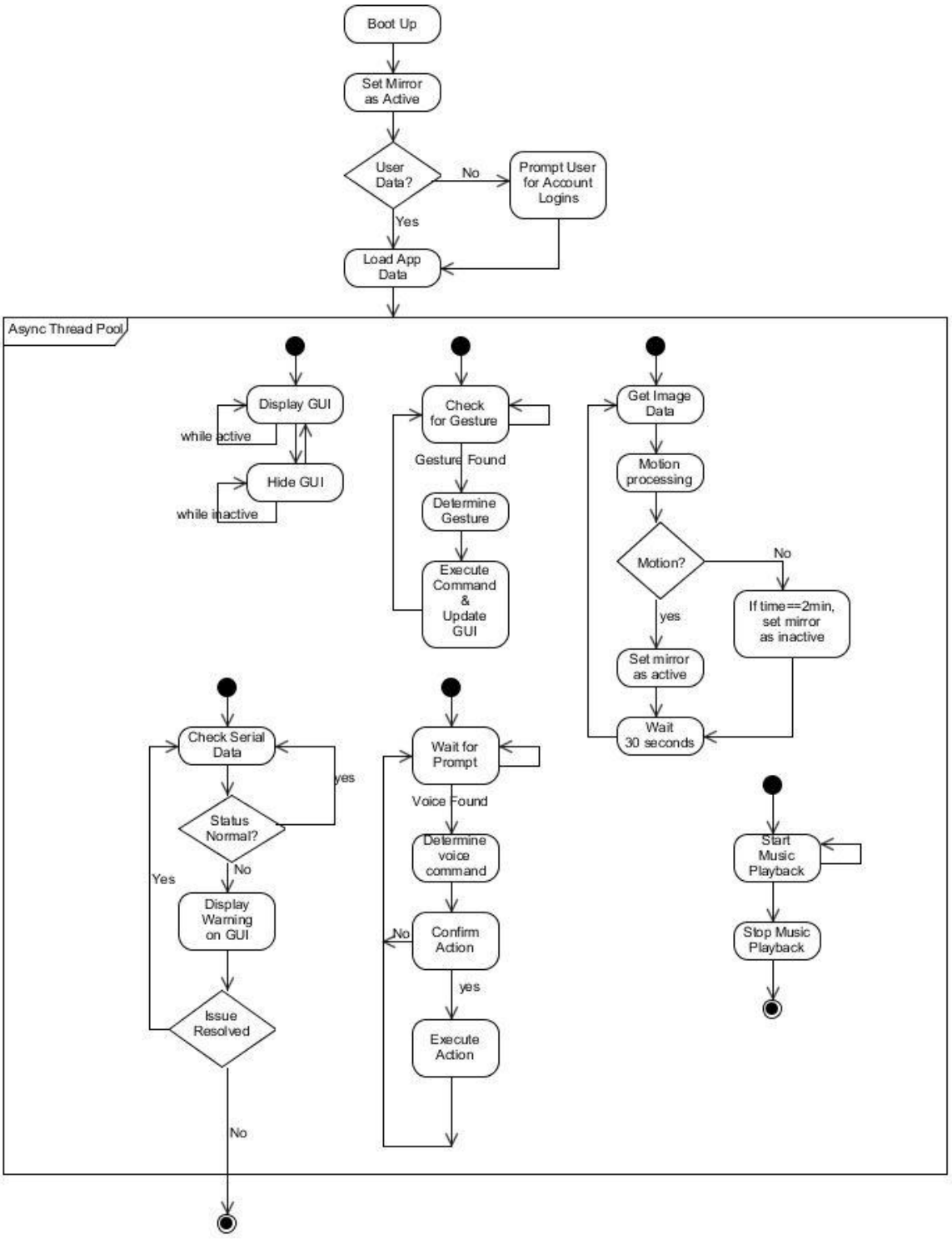

Figure 5.2.4 – Central PC State Diagram

Motion detection was the final computing intensive algorithm that was running on the central PC. Previously mentioned in section 4.2.2, the algorithm can be seen in 5.2.5 below. The algorithm was designed to run every 30 seconds in the applications asynchronous thread pool as it searched for the user. The basic flow of the algorithm

involved comparing two images pixel to pixel for intensity differences. If the intensity threshold was exceeded in both change and quantity, motion was determined to be present. Looking at the figure, as soon as the system is booted up, the webcam will begin checking the images. The current image is captured and a check is performed to verify that there is a reference image to compare to. If there isn't an image, the first image taken will be set as the reference image and a new "current" image will be taken by the webcam. In the Windows 8 API, a BitmapDecoder object must be set up to get pixel data from an image frame. Then calling the GetPixelDataAsync provides the image data of the pixels into the ProcessPixelArray [58]. Thus, multiple for-loops are performed to then compare pixel-to-pixel each of the two images. The pixels are compared in terms of intensity and color since depending on the format; each pixel will have its own value that is comparable. A threshold value was set up during coding that will adjust how sensitive the motion detection is. This threshold value will be compared against the numeric difference between the two pixels. If the difference is greater than the threshold, the coordinates of that pixel will be set as "movement". If not, the pixel coordinate will be set as "no movement". This will repeat from the "capture image" block in Figure 5.2.5 until all the pixels have been compared and checked. Here the total number of pixels marked as "movement" will be totaled up and compared against another threshold in order to limit false "movement" due to noise.

Assuming that the number of "movement" pixels is greater than the threshold, the motion variable will be set to true and the waiting variable set to zero. Once motion is found, the system checks whether the mirror is set to "active". If yes, the system waits 30 seconds and then repeats the whole process from the beginning. If no, the mirror is set as active, waits 5 seconds, and then repeats the process. On the other hand, if no motion is found, the same question is asked. If the mirror is active, then the system waits 30 seconds, increments the waiting variable by one, and then checks the value of the waiting variable. Once the waiting variable is 4 or greater, that means at least 2 minutes have passed since motion was last found so it is safe to say the user is no longer present. This then prompts the system to go into "not active" mode. Otherwise, in either choice, the webcam will continue to check for motion. Finally, once the mirror is inactive and no motion is being found, the default process is to wait 5 seconds then repeat the whole algorithm again.

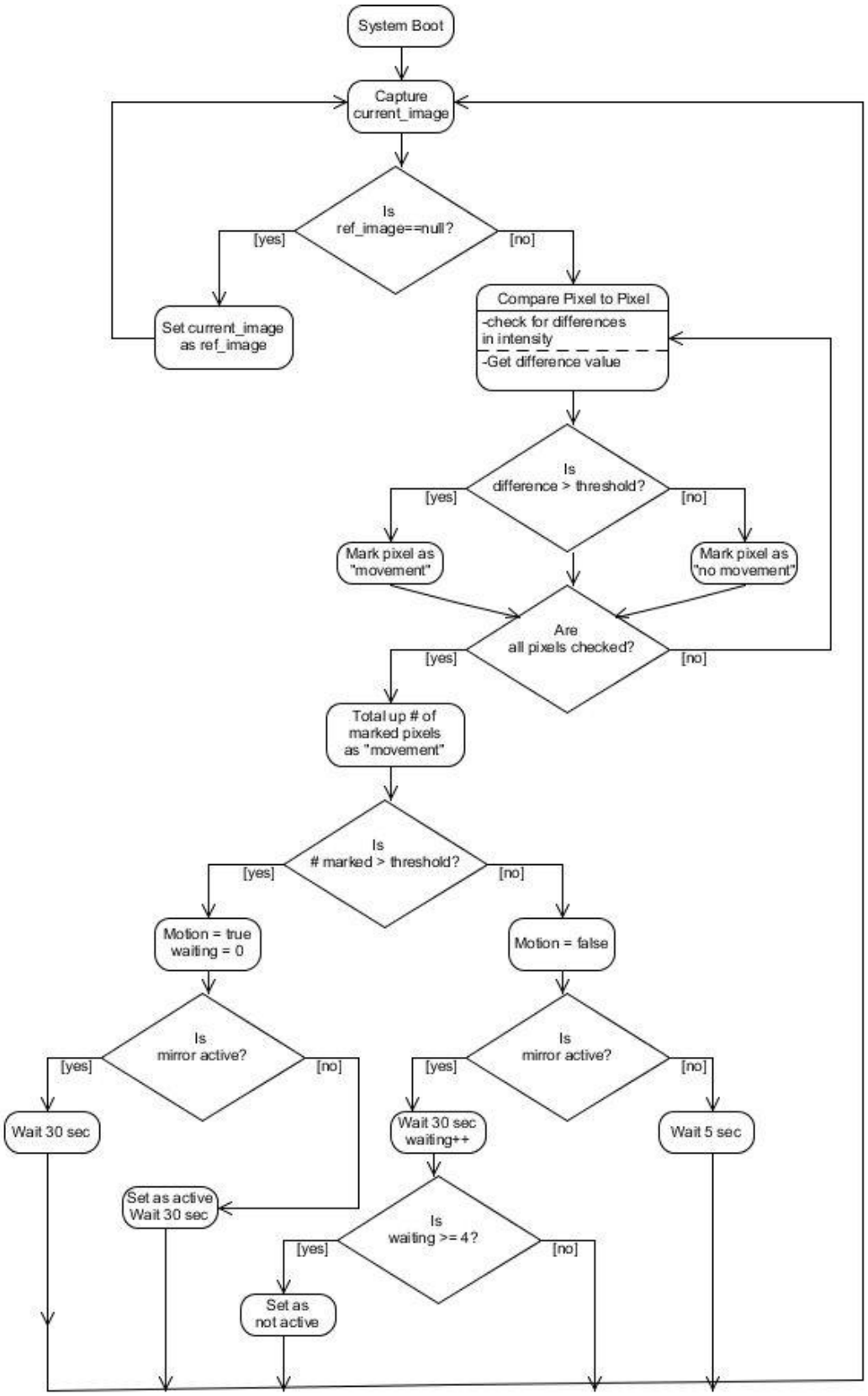

Figure 5.2.5 – Motion Detection State Diagram

The voice recognition was another important software aspect. When the user performs the correct gesture, the voice controller leaves the default state and moves to record 5 seconds of audio from the user. This audio is saved to a WAV format and uploaded to the AT&T Speech server. The server translates the voice to text and returns what it hypothesizes the user said. If the returned text is empty or doesn't contain any of the command key words, then the program exits back to the default voice state. When a correct command is said, the program moves on to determine what keyword was stated. The key words include "tweet", "task", "play", "pause", "next", "previous", and "stop". The keyword "tweet" involves posting a tweet to Twitter. The keyword "task" involves adding a new item to the user's to-do list. The other keywords are all commands to control the music. So, if the user uses a Twitter or to-do list keyword, they are next prompted to say whatever tweet or task they wish to add. Once this is converted to text, the text is displayed to the user in order for them to either confirm with a "yes" or deny with a "no" if the text translated correctly. If it isn't correct, the action is canceled and the system returns to the default voice state. If it is correct, the action is performed. Regarding the music commands, they are processed similarly except there is no confirmation. The keyword located in the voice command text is executed directly such as "play music". This statement will instantly begin playing music located on the smart mirror hard-drive. The process can be seen in Figure 5.2.6 below.

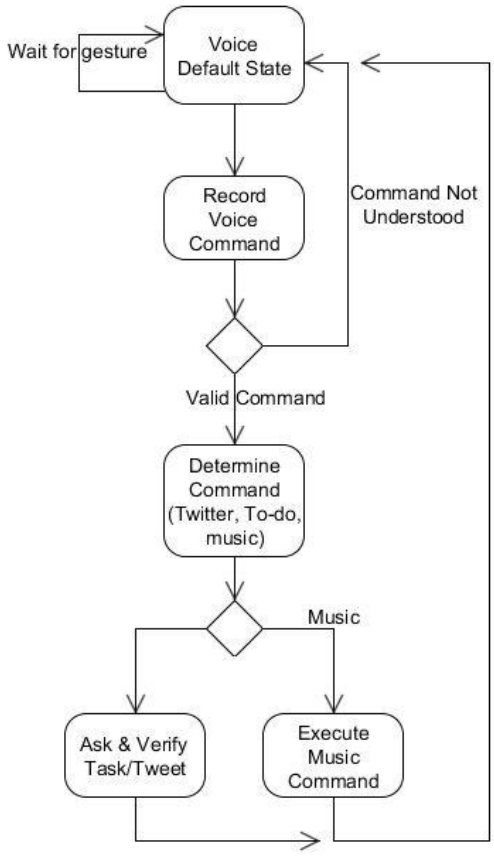

Figure 5.2.6 – Voice Recognition State Diagram

The MCU code was important because it is what communicated the temperature data to the central PC. The MCU on setup sets the appropriate analog pins and opens a serial connection with a 9600 baud. Once completed, the program enters the main loop. Here in the main loop, first the analog temperature data is sampled from the assigned pin. In order to prevent extraneous and inaccurate data, the temperature sensor is sampled 8 times. This temperature data is then averaged and converted to Celsius. Next, the humidity is sampled multiple times as well for an accurate reading. The humidity is then put through a series of conversions that takes the ADC data and puts into the correct percentage format. This process is repeated after a short delay. The control PC checks the temperature and humidity data every 5 seconds. The control PC sends a request character serially to the MCU and the MCU gets interrupted to return the most recent temperature and humidity data back to the PC. This process is represented in a state diagram in Figure 5.2.7 below.

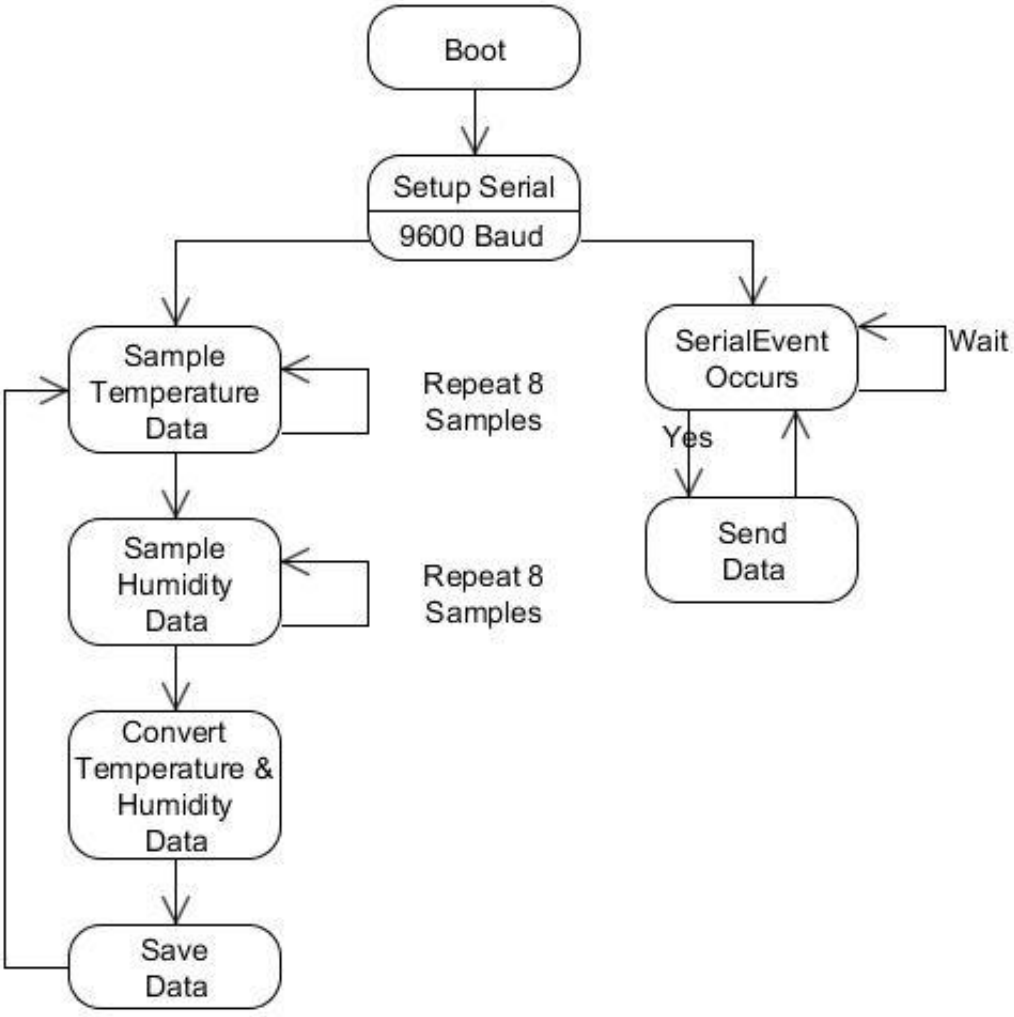

Figure 5.2.7 – MCU State Diagram

5.3 Housing

For the housing of the entire mirror there were many things which we needed to consider. First was to determine the best material to build the case. Second we had to determine the dimensions of the entire casing and where to hold the television, mirror, CPU, webcam, speakers, and leap motion controller. Once we figured out the material, the dimensions, and sizes we then planned the most efficient way to build and compile it all together into one finished product.

There were many materials to choose from in building the case. The factors that determined what we chose were mainly the cost of the material and also if the material could be damaged potentially by the high humidity in a bathroom setting. Out of all the materials there were to choose from the most crucial requirements were weight and protection of the equipment. The weight of the material couldn't be too heavy but also had to be strong enough to protect the television and keep the two way mirror from getting damaged.

Since a television was placed vertically into an encasing, a wooden frame was used for the front of the mirror. This frame held the television and the webcam in place and had an insert that kept the mirror firmly against the television while keeping it all stationary. It was beneficial to use two separate cases, one for the mirror and the other for the PC. The reason for this was that if the PC was in the same case as the mirror and television the case would be very bulky. Also if there were any problems or issues with any of the hardware it was easier to diagnose due to the fact that it was easier to access the PC if it was not behind a mirror ideally placed in a bathroom setting.

The basic outline of the "Smart Mirror" started with the reflective glass (two way mirror) facing the user. Behind the glass was the television which displayed all the applications of the smart mirror. Connected to the television was the PC which had the webcam, speakers, leap motion and MCU linked to the humidity and temperature sensors connected to it. In addition to this a webcam was placed on the top of the mirror inside the encasing as well. There were also two speakers in the top corners of the border. These all needed to be casted into the display without affecting the mirror or television. The various components were glued down for protection and safety. On the bottom of the border the leap motion controller was attached in its own bracket. It was glued down so that it could be accurately calibrated.

A sketch of what the smart mirror looked like is shown from Figure 5.3.1 to Figure 5.3.3. These three figures show the various angles for the smart mirror design. Figure 5.3.1 shows the front view of the mirror displaying the locations of the webcam, speakers, and Leap Motion controller. Figure 5.3.2 shows the side angle of the smart mirror. Here the placement of the hardware components can be more easily observed. It is clearly shown how the two-way mirror and television are secured inside the housing. The brackets for the television allow for a firm hold while still providing easy access to the backside of the television components. Finally, in Figure 5.3.3 the shelving and brackets used to hold the components within the encasing are shown. The speakers and webcam sat on a custom

shelf within their designated cut-out locations and were glued down to prevent them from moving around.

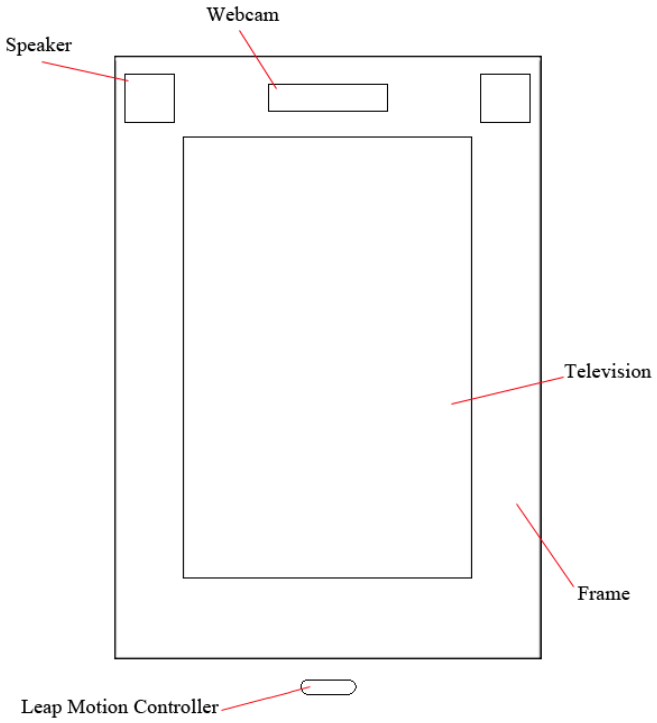

Figure 5.3.1 – Smart Mirror Design Layout

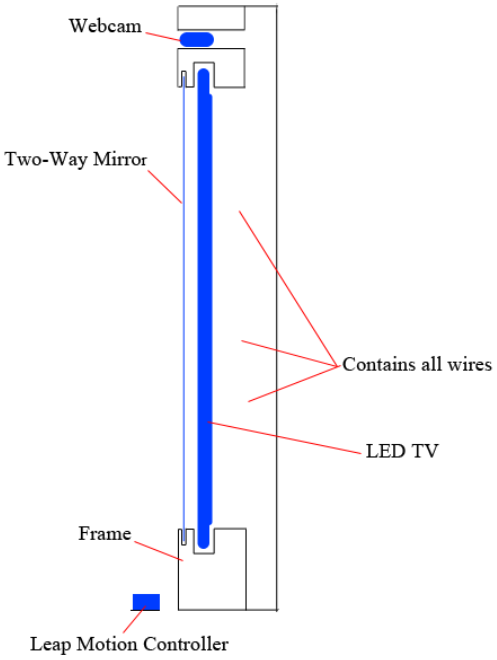

Figure 5.3.2 – Smart Mirror Side Design Layout

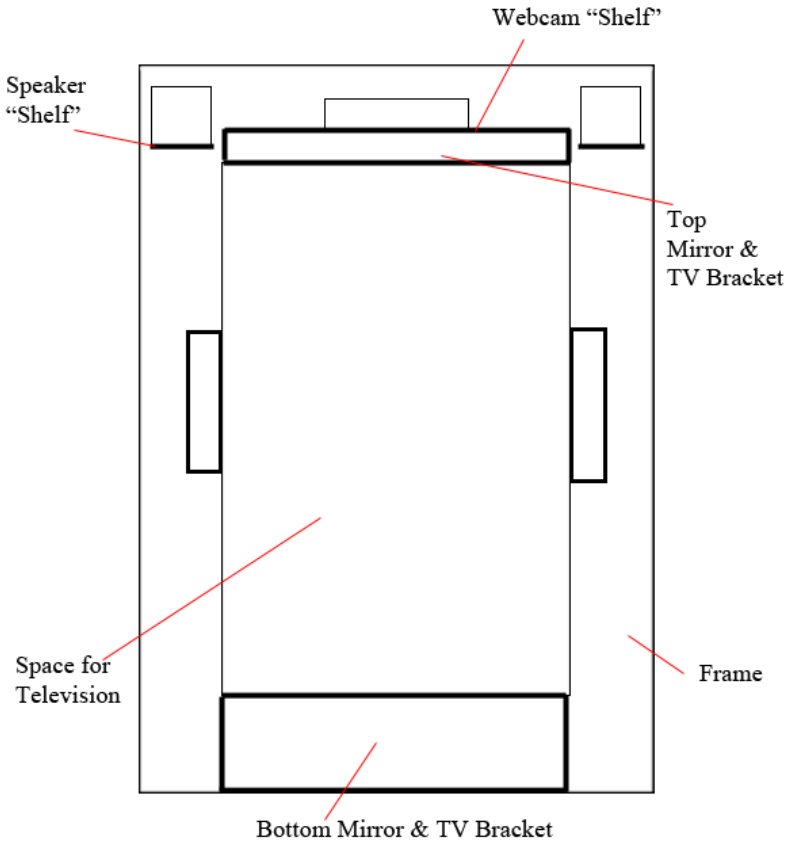

Figure 5.3.3 – Smart Mirror Backside Design Layout

6.0 Project Prototype Construction

6.1 PC

For construction of the central PC, the prototyping is straight forward. This part of the smart mirror system requires hardware to be selected based off the specifications of the other elements that will make use of the PC. There are six essential pieces to building a working computer with various models of each part available for use. The selection of the motherboard provides the specs that all the other parts will follow in order to ensure compatibility.

6.1.1 Parts List

Table 6.1.1.1 displayed below lists the six essentials hardware parts to building a functional computer plus the addition of a wireless adapter since the central PC must be wireless. The total cost of all the parts for the central PC is calculated by selection of the

cheapest seller, NewEgg. Also considering the price of shipping, NewEgg offers free two-day shipping, which makes it the clear option.

Component	Model	Seller	Price
Antec 380W PSU	EA-380D Green	NewEgg	$44.99
Adata 64GB SSD	ASP900S3-64GM-C	NewEgg	$74.99
ASRock Intel Mini ITX Motherboard	H87M-ITX	NewEgg	$94.99
G.Skill 4BG DDR3 1600 RAM	F3-12800CL9D-4GBRL	NewEgg	$44.99
Intel Core i3-4340 3.6GHz CPU	BX80646I34340	NewEgg	$159.99
Cooler Master Elite 130 Mini-ITX Case	RC-130-KKN1	NewEgg	$49.99
TP-Link IEEE 802.11 a/b/g/n	TL-WDN4800	NewEgg	$42.99
		Total	$512.93

Table 6.1.1.1 – Central PC Hardware Parts

6.1.2 Parts Integration

Once the parts have been acquired, the steps for putting them all together involve either snapping them together or screwing them together. The group's past experience with building their own computers eliminates the need for taking time to research how to do it correctly.

Assembly began first with clearing an area for a static-free environment and the assembler grounding their body. The motherboard was laid out, the LGA 1150 was unlatched using the small metal bar, and then the plastic cover was removed from the CPU socket. Here, the i3 processor was gently placed, making sure to line up the pins correctly. The metal socket brace which was removed before, was now placed back to down to hold the processor in place. The metal bar was pressed back down and locked to secure the newly mounted processor. Next, using the thermal compound, a tiny pinch was applied to the surface of the newly mounted processor metal surface. Using a sturdy, thin card, such as a credit card, the compound was spread out evenly over the processor. The layer of the compound should be very thin and evenly applied to the surface. Any excess was wiped off. This thermal compound was very important because it assists the heat sink in keeping the CPU cool and drawing heat away. This was why the compound should be applied evenly because it wouldn't be ideal if some areas of the processor weren't getting cooled as efficiently. Now at this stage, the heatsink was mounted on top of the processor, with the "legs" of the heatsink lining up with the screw holes on the motherboard. A heatsink backplate was aligned with the screw holes on the backside of the motherboard. Screws were then screwed in to secure the heatsink to the motherboard and provide a solid connection between the processor and the heatsink.

Once completed, the most difficult part of the assembly was done. Next, the motherboard was aligned with the screw mounts in the case and screwed into place. The final parts left can be integrated in any order. Take the PSU and screw it into the appropriate place in the case. The PSU has several power cables to power the various components but to avoid hassle, those are best left to last. Unhinge the RAM slots, and the RAM was gently

pressed into each of their 240-pin slots until they click. The SSD was mounted into the HHD slot using the bracket provided and screws. Next, the SATA cable was connected into the SSD and then into the SATA slot on the motherboard. In this build, there was a PCI-e wireless adapter. This adapter was placed into the PCI-e slot and secured by a single screw. Finally, the cables from the PSU could be pulled into their appropriate slots. The 4-pin CPU connector was inserted on the motherboard, near the CPU. The 24-pin motherboard power connector was put in after and finally the connector for the SSD was plugged in. This completed the assembly of the central computer with only the installation of Windows 8 left.

6.2 Temperature Regulation System

6.2.1 Parts List

Table 6.2.1.1 displayed below lists the specific parts that were needed to create the humidity and temperature sensor circuit on the printed PCB. Each of these individual pieces was ordered from certified circuit suppliers. These pieces were combined with printed circuit board through the process of soldering and other steps.

Part Name	**Price per unit**	**Quantity Ordered**	**Total price**
LM35D Temperature Sensor	$4.96	2	$9.92
P-14 Humidity Sensor	$24.28	1	$24.28
Various Resistors (Extras)	$16.45	Various	$16.45
TL084P	$0.99	1	$0.99
2N221	$9.10	2	$18.20
Total			$69.84

Table 6.2.1.1 – Part List for the Temperature/Humidity PCB

6.2.2 Assembly

For the PCB design, the circuit was printed by a professional company called Advanced Circuits. Once PCB design was done, we still needed to solder components on the circuit board. The first step of soldering was surface preparation. What was done first was clean the surface by using Scotch Brite pads. After that, we used solvent (acetone) to wash off chemical contamination from the surface. The second step was to place all necessary components on the board. We started with the smallest components first such as resistors and diodes, and then moved on to bigger components such as the sensors.

The third step was applying heat. We placed the tip of the iron against both the component lead and the board. It usually took a second or a little bit more to solder the components depending on thickness of the components. A very important thing that we

needed to keep in mind was that if it started bubbling, we needed to stop heating and allow the material to cool down. Once it's cooled down, we tried to apply heat again cautiously.

The fourth step was to apply solder to the joint after the solder pad and component lead reached high enough temperature. We stopped adding solder after the surface pad is coated.

The last step was the inspection of the joint and cleanup. If there are any cold joints, then we had to reheat them, apply a little bit of solder, and then let it cool for a while [66]. After that, we would clean the entire excess flux residue from the circuit board.

Another important aspect of assembly besides soldering was the calibration of the moisture control switch. The calibration could be done by dissolving a small amount of salt in a cup of water which will cause the moisture level of the room to rise up to approximately 90%. After that, we set up the second sensor to get a voltage of around 2V [67]. Finally, once all the components were soldered, using a voltmeter, each component was tested to make sure it was working correctly.

6.3 Custom MCU

6.3.1 Parts List

Table 6.3.1.1 displayed below lists the specific parts that were needed to create the custom Arduino on the printed PCB. Each of these individual pieces was obtained from de-soldering them from the Arduino Uno board for both time and financial reasons. These pieces were them re-soldered to our custom Arduino printed board through the process of soldering and other steps.

Part Name	**Price per unit**	**Quantity needed**	**Total price**
Arduino Uno Board	$35.00	1	$35.00
USB Plug	$0.00	1	$0.00
Atmel Mega16U2	$0.00	1	$0.00
ATMega 328p	$0.00	1	$0.00
16 Mhz Crystal Clock	$0.00	1	$0.00
SPK16	$0.00	1	$0.00
09T1 R5R	$0.00	1	$0.00
Resistor/Capacitor	$0.00	(various)	$0.00
Total			$35.00

Table 6.3.1.1 – Part List for the Custom Arduino Uno

6.3.2 Assembly

For the MCU design, the circuit was printed by Advanced Circuits using the customized Eagle file provided from the Arduino website. Once board design was printed, we soldered the components on a circuit board that we needed. Some of the components were too small to solder ourselves and we went to Quality Manufacturing Services (QMS) for assistance.

6.4 Housing

6.4.1 Parts List

Table 6.4.1.1 displayed below lists the materials required for building the smart mirror housing. The main materials were the wood, paint, the nails, and the two-way mirror. Each item came within budget and was purchased from Home Depot. Only exception was the two-way mirror which was purchased from an online company called Reflection Products. Table 6.4.1.2 below Table 6.4.1.1 lists the hardware components that were purchased that were housed inside the encasing. These items included the webcam, television, speakers, and the Leap Motion controller.

Part Name	**Price per unit**	**Quantity**	**Seller**	**Total**
2”x4”x8” Premium Kiln Dried Whitewood Stud	$6.56	5	HomeDepot	$32.80
Grip-Rite 9x3.25in Vinyl-Coated Steel Smooth Sink Nails	$10.48	1	HomeDepot	$10.48
Paint	$8.99	1	HomeDepot	$8.99
Two-Way Mirror 27.5” x 15.7” Sheet	$99.00	1	reflectionproducts.com	$68.91
			Total	$121.18

Table 6.4.1.1 – Parts list for Smart Mirror Housing

Component	Model	Seller	Price
Insignia 32” LED – 720p HDTV	Insignia153 3234 Class	BestBuy	$191.69
Logitech C920 HD Pro Webcam	960-000764	NewEgg	$89.99
ARCTIC S111 Stereo Speakers	ORACO-SP001-GBA01	Amazon	$20.34
Leap Motion Controller	LM-010	Leapmotion.com	$79.99
		Total	$382.01

Table 6.4.1.2 – Hardware Components list for Smart Mirror Housing

6.4.2 Assembly

The frame of the encasing was built using 2x4 pieces of wood. It was roughly 30x18 inches. The wood was connected using nails. This was the best way to ensure stability and also facilitate the process in terms of time and work. We built the rectangular border first. Once the border was built, we built the three dimensional frame. This included the bracket shelf for the leap motion controller to be held. Once the leap motion controller bracket was created and secured onto the front frame of the mirror then the holes were made to hold the Speakers and webcam. The speakers that were used are 3’x3’ speakers which fit directly into the square cut out of the wood. There were small “brackets” behind the frame that were made to hold all the pieces in place such as the HDTV. This can be seen in the concept drawing in Figure 6.4.2.1 below. A rectangle was cut out from the top center of the frame for the webcam. Once all these pieces were added correctly they were taken out and the outer casing was sanded, painted and finished. This gave it a nice look on the outside. After that the equipment was added and secured onto the frame using glue. The two-way mirror was then be added and secured to the HDTV screen with glue thus completing the front view of the “smart mirror.”

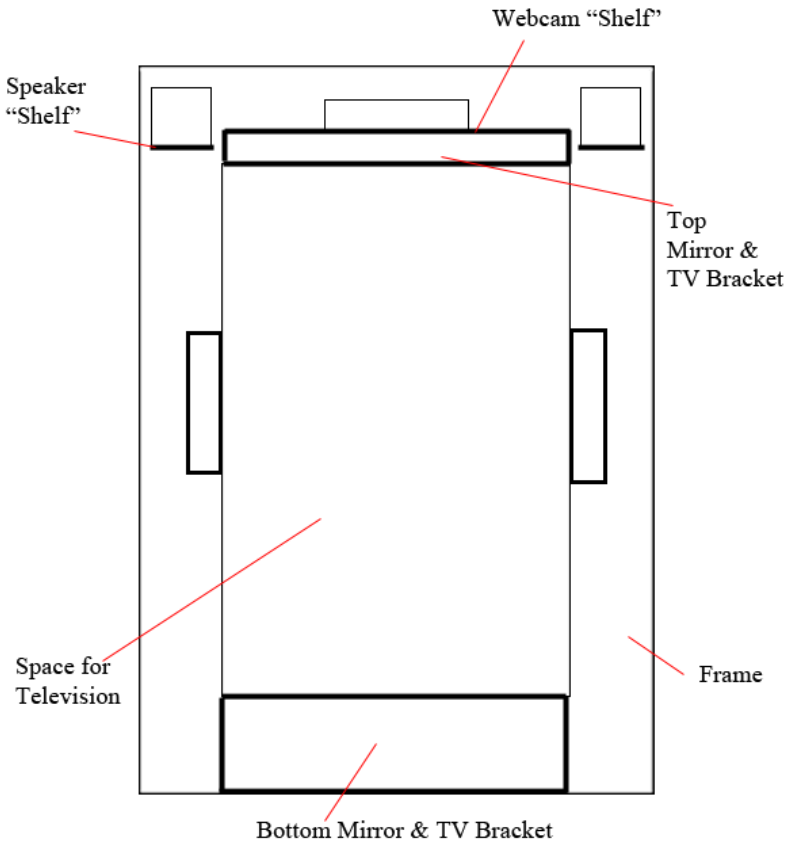

Figure 6.4.2.1 – Smart Mirror Backside Design Layout

After the two way mirror was placed on the TV, the television was placed immediately into the frame, leaving as little to no space as possible. Once the television was placed in its bracket, it will be secured using tape. After the television was secured, all the wires were connected and we moved on to the testing phase. They were connected to the power source and central PC was located below it.

7.0 Project Prototype Testing

7.1 Hardware and Software

The process of testing is what ensures that the final product performs to the specifications and requirements stated before the design phase. Thorough testing is important to provide evidence that the design is either working or contains bugs that must be addressed before leaving the hands of the developers. The hardware testing will involve both testing that the physical hardware works but also that the hardware is working as it should within the system with the software. This ensures that the hardware and code are acting as designed. Each part of the system will go through a unit test which primarily tests the functionality that the part should be doing as well as possible abnormal conditions.

7.1.1 Unit Test – Temperature/Humidity Sensors

The testing of the temperature and humidity was specifically about how the sensors react to their designed thresholds. The sensors must perform accurately in order to be reliable for a temperature system. The tests completed for the temperature sensor can be seen in Table 7.1.1.1 below. The tests completed for the humidity sensor can be seen in Table 7.1.1.2 below.

Test	**Steps**	**Expected Results**
Determine accuracy	Obtain results from low temperatures (7.2 °C) Obtain results from room temperatures (~ 21 °C) Obtain results from high temperatures (32 °C) Obtain results in the area of high humidity (above 90%)	+/- 1 °C error

Table 7.1.1.1 – Temperature Sensor Tests

Test	**Steps**	**Expected Results**
Determine accuracy	Obtain results in the area of low humidity (below 40%) Obtain results in the area of standarand humidity (40%-60%) Obtain results in the area of high humidity (below 40%) Obtain results from high temperatures (32 °C)	+/- 1.5% error

Table 7.1.1.2 – Temperature Humidity Tests

7.1.2 Unit Test – Microcontroller: Signal Control

The testing of the custom microcontroller signal control is specifically about how it interacts with the temperature regulation system PCB. The system was controlled by analog pins, so the unit test showed that the pins were being activated and contain the correct value. These tests can be seen in Table 7.1.2.1 below.

Test	Steps	Expected Results
Specific Pin	Set desired pin to the highest active setting	The pin should display a value on the sample serial output
Pin Value	Vary the value of the pin by connecting it to the temperature PCB	The pin value should transmit the correct value serially to the PC
Code Behavior	Simulate values coming from the temperature and humidity sensors.	Should serially transmit to the central PC the temperature values

Table 7.1.2.1 – Custom MCU: Signal Control Tests

7.1.3 Unit Test – Webcam: Motion Detection

The testing of the webcam goes hand-in-hand with the motion detection algorithm. This unit test confirmed that the webcam was interacting with the code correctly and that the code was obtaining the correct image pixel data to determine motion. This test also tested the sensitivity of the motion detection software. These tests can be seen in Table 7.1.3.1 below

Test	Steps	Expected Results
Detects Motion	Let it run for 1 minute to calibrate, then have user walk into view of the camera	Webcam should detect motion and output a true value
Detects No Motion	After passing the detect motion test, user will leave camera view.	After 2 minutes, the webcam should confirm there is no more motion and output a false value
Determine Sensitivity	Change lighting in room (vary from very dark to very bright) Have user move in front of camera under each brightness setting	The physical lighting change shouldn't activate the mirror. The motion of the user should be recognized in the low light to high light situations. Not expected to pass very dark tests

Table 7.1.3.1 – Motion Detection Tests

7.1.4 Unit Test – Microphone: Voice Recognition

The testing of the microphone involved use of the webcams microphone. The unit was setup to look for specific words spoken in order to test the accuracy. Not only was accuracy important, but the sensitivity as well. The tests for the microphone and voice recognition can be found in Table 7.1.4.1 below.

Tests	**Steps**	**Expected Results**
Determine accuracy	Say different commands and observe if the system recognizes them	The voice recognition should be able to recognize all the commands in quiet atmospheres
Determine speaker-independency	Let another person say the same commands and observe if the system recognizes them	The voice recognition is speaker-independent and hence should be able to recognize different voices.
Sensitivity	Test how sensitive voice recognition is by saying commands with different voice levels	The voice recognition should be able to recognize commands most mid-range to loud voices

Table 7.1.4.1 – Voice Recognition Tests

7.1.5 Unit Test – Leap Motion: Gesture Recognition

The Leap Motion controller was in charge of recognizing the user's gestures for interaction within the user interface. This unit test confirmed that the controller is seeing the gestures correctly and in the correct spot. These tests can be seen in Table 7.1.5.1 below.

Test	**Steps**	**Expected Results**
Detects Gestures	Perform left and right swipe Perform up and down swipe Perform point and hold Perform clockwise circle	The type of gesture in the output code matches that being performed
Detect Region Specific Gestures	Perform horizontal swipe left of screen origin Perform horizontal swipe right of screen origin Perform horizontal swipe across screen origin	The swipes left and right of the screen origin should be recognized as a swipe in that region The swipe across the origin shouldn't recognized for either the left or right side
Determine Sensitivity	Change lighting in room (vary from very dark to very bright)	The gesture of the user should be recognized in the low light to high light situations.

Table 7.1.5.1 – Leap Motion Gesture Tests

7.2 User Interface

The user interface was a very intricate part of the design and had to fulfill requirements on both a functional and nonfunctional level. The functional requirements and specifications were the specific features and design elements set for the smart mirror. These elements were objective and had to obtain their goal to be considered as a "passing" test. On the other hand, the user interface had to also meet a standard of "quality" by passing a series of non-functional tests which tend to be more subjective than objective. The determination of pass or fail was more up to the thoughts, opinions, and decisions of the developer.

7.2.1 Functional

The functional tests of the user interface were the objective actions that must occur for the smart mirror to maintain its user interface integrity. The functional tests covered actions of the individual applications, UI interactions, voice recognition, and various procedures relating to the user. The functional tests were broken into two tables. Table 7.2.1.1 below provides details on the tests and results for assorted functions. Table 7.2.1.2 below the other table provides details on the tests and results on the gesture focused interaction Table 7.2.1.3 provides details on tests directly related to the voice control.

Test	**Steps**	**Expected Result**
Applications load data	Activate mirror, then switch all applications into normal mode Observe each app individually	The applications should load data from their specific server and display it correctly
GUI Visibility	Activate the mirror so that GUI is brought up on the screen behind the mirror	The GUI should be easily seen through the mirror and any part that is "black" in the GUI should be hidden
Mirror Turns On and Off Automatically	When the mirror is not active, user walks in front of camera When the mirror is active, user walks out of view of the camera and waits	The mirror should become active once the camera detects the user's movement The mirror should become inactive after 2 minutes of no movement
Displays Temperature Warning	Force code to issue warning that the temperature/humidity in the room is dangerously high	The GUI should show a warning message telling the user of the issue
Changing App Data	Start the mirror like normal then activate the data change button and change the weather zip code	The application should load weather relevant to the zipcode entered and should remember the zipcode when booted again

Table 7.2.1.1 – Assorted Functional Smart Mirror Test

Test	Steps	Expected Result
Left and Right Side Swipe Mode-Change Gestures	On the side left of the screen origin, perform a horizontal swipe to the right and to the left Repeat for the side right of the screen origin	On the left side, the right swipe should change the 3 apps from minimized to normal mode. The left swipe should return them to being minimized. On the right side, the left swipe should change the 3 apps from minimized to normal mode. The right swipe should return them to being minimized.
Point and Hold Mode-Change Gesture	Point and hold an app, wait, and then swipe up Repeat for each of the 6 apps	When the app is interacted with, it should switch from normal mode to expanded mode. When the swipe is performed on the app, it should change back to normal mode
Circle Refresh Gesture	Perform the circle gesture anywhere on the screen	The apps should all refresh their data from their sources
Circle Activate Voice	Perform the counter-clockwise circle gesture anywhere on the screen	The mirror should prompt the user to begin dictation

Table 7.2.1.2 – Gesture-based Functional Smart Mirror Tests

Test	Steps	Expected Result
Control Music	Dictate to the mirror to play and then stop music playback	The voice commands should perform the specified action at least 75% of times attempted
Post a Tweet	Dictate to the mirror to post a tweet. Say a tweet and confirm it with “yes”	The voice commands should perform the specified action at least 75% of times attempted
Add a new Task	Dictate to the mirror to post a new task. Say a task and confirm it with “yes”	The voice commands should perform the specified action at least 75% of times attempted
Say Nothing	Activate the mirror to listen for voice input and say nothing	The mirror should say nothing was heard and exit without issue

Table 7.2.1.3 – Voice-based Functional Smart Mirror Tests

7.2.2 Non-Functional

The non-functional tests of the user interface were a combination of objective actions with more subjective basis. They determined the true quality and have just as much of an impact on the user as the functional actions do. Tables 7.2.2.1and 7.2.2.2 below provide details for each test and what the expected outcome were to be considered "passing".

Test	**Steps**	**Expected Result**
Speaker sound level	Play notification sound and a sample music file. Have three people listen and provide feedback	All users should agree that the sound level is appropriate and audible
Gesture Recognition Percentage	Have three different users perform 15 various, valid, gestures while interacting with the mirror UI	Out of the 45 gestures performed, a 90% or greater success rate should be met
Boot/Shut-down time	Boot-up and shutoff the system 5 times each	The average startup time must be 1 minute or less The average boot-down time should be under 30 seconds
Application Refresh time	Perform the circle gesture to manually refresh all the apps	The applications should all refresh their data in under 5 seconds
Voice Recognition Percentage	Have three different users perform 15 various, valid, voice commands while interacting with the mirror	Out of the 45 voice commands, a 75% or greater success rate should be met

Table 7.2.2.1 – Non-Functional Smart Mirror Tests (Table 1 of 2)

Test	Steps	Expected Result
UI Intuitiveness	Have 3 first-time users interact with the mirror with minimal gesture and command instruction	The users should adapt and be able to discover/use all aspects of the UI in a timely manner
Motion Detection Time	Perform 10 instances where the mirror is inactive, then the user walks in front of the mirror. Then the user leaves and lets the mirror become inactive	The mirror should become active within 10 seconds of the user stepping in front of the mirror The mirror should then become inactive within 2 minutes
TV Brightness	Observe the GUI produced by the TV through the mirror	Only the parts of the GUI that aren't a black color should be visible. All other parts of the screen should be "mirror" and unable to see the TV through it
Light Brightness	Observe the room's lighting effect on the mirror's screen and GUI	Each light level should not hinder the visibility of the GUI or "mirror"

Table 7.2.2.2 – Non-Functional Smart Mirror Tests (Table 2 of 2)

7.3 Integration

7.3.1 System Test

The system testing were the final tests that encompassed all aspects in the unified system. Various functional, non-functional, and unit tests were repeated within each of the system tests. These were the closest tests to how the user will be actually using the smart mirror. The only difference that sets these tests apart from the unit tests was that they aren't specifically being tested; the components were expected to work correctly as part of the system. Table 7.3.1.1 below lists the tests being performed.

Test	Steps	Expected Result
Auto-On and Auto-off	User walks in front of mirror to be detected User continues to walk away from the mirror and then returns several times at different intervals	The mirror should behave as expected. Mirror should stay active as long as there is movement every two minutes.
General Bathroom Temperature	Run shower on hottest setting for 10 minutes with door open Wait for bathroom to dry and repeat with door closed Use debug interface to observe temperature and humidity sensor data	The mirror should still be useable in terms of the motion detection, leap motion, and general interaction. If thresholds are broken, system should react accordingly
Playing Music	The user should use the voice command to play music The user should interact with other apps while the music playing The user should stop and play the music with voice commands	The music should play seamlessly in the background switching to a new song when the previous is completed The mirror should stop and play the music correctly whenever the voice controls are being used
Various Gestures	The user should perform the gestures setup to be used with the smart mirror The angle of the swipes and other gestures should be varied The speed of the swipes and other gestures should be varied	The gestures should be recognized correctly within a margin of error. Vertical swipes shouldn't be recognized as horizontal swipes or visa-versa. The speed of the swipe should have minimal effect on how the well the gestures are recognized

Table 7.3.1.1 – Full System Smart Mirror Tests

8.0 Administrative Content

8.1 Milestone Discussion

Designing a project that emphasized time management and multitasking, our team made certain to have a milestone plan on when each part of the project must be completed. During senior design I, this phase included a majority of the research and design phases. These were expected to be completed before senior design II where the prototyping and testing were executed.

The research in the first semester allowed for the team to learn about the different components in the design and get familiar with what we actually need to acquire. The design evolved along with the research to fit what is actually possible and what needs to be done to stick to the timeline and budget. Although there was planned prototyping, the prototyping that occurred was purely software. The minimal planned physical prototyping was pushed to Senior Design II. Besides that, the schedule was heavily followed, with research being the focus of the first couple months and design rounded out the end of the semester. These patterns can be seen in Table 8.1.1 below where the specifics are laid out along with the dates.

Task	Sept 1	Sept 15	Oct 1	Oct 15	Nov 1	Nov15	Dec 1	Dec 15
Voice Software								
Two-Way Mirror								
Application Program								
Temperature Sensors								
Humidity Sensors								
GUI								
Housing								
MCU								
Gesture Recognition								
Central PC								

Legend	
Research	
Design	
Prototyping	

Table 8.1.1 – Senior Design I Schedule

For Senior Design II, the main focus was prototyping and testing. All the design was done and only has to be followed through. The physical pieces of the central PC, PCB, housing, and other parts were acquired. The software was coded piece by piece early on in the semester to leave plenty of time for testing. During this time as well, the PCB was be assembled as well as the housing. This included the MCU as well, which was delayed to later in the semester due to complications. This left time to fix any design flaws and reorder parts if necessary. Less than two months were left for unit and full system testing to make sure that the final product had minimal bugs. The specifics of this schedule are in Table 8.1.2 below.

Task	Jan 1	Jan 15	Feb 1	Feb 15	March 1	March 15	April 1	April 15
Housing								
Temperature and Humidity PCB								
Custom MCU								
Central PC Applications								
Completed Mirror								

Legend	
Ordering	
Prototyping	
Testing	
Debugging	

Table 8.1.2 – Senior Design II Schedule

8.2 Budget and Finance Discussion

The budget for the smart mirror project was developed early on in the project process. The original budget was setup with slight overestimates in each category to allow for movement of costs around to different components once proper research had been conducted. The goal was to stay around or under the price of a standard laptop since that keeps the mirror in the financial range of most people. Once official purchases were

made, adjustments were necessary, but the budget reallocations due to the decisions on specific materials and items were still kept within budget. Table 8.2.1 represents the original budget setup and the now adjusted budget based on the product purchases. Some prices couldn't be set until quotes were ordered from the manufacturer but the parts that had been confirmed were updated. This can be considered a bill of sale not just a budget reallocation since these prices are now budgeted to cover shipping and taxes. Looking at Table 8.2.1 below, the original budget clearly overestimated some particular items and underestimated others. The "change" column directly identifies which categories were altered, either in a positive, negative, or neutral manner. There were more budget cuts than budget growth except when it came to the cost of the PCBs and a couple other categories. Overall, the new final budget represents a much tighter or more accurate representation about what the smart mirror cost. The project came close to staying within the budget set despite the changes that came up. With all consumer geared projects, price was always a factor. It benefited this project and its goals that the new budget was overall almost exactly on budget except by a $1.55.

Item	**Original Budget**	**Final Budget**	**Change(+/-)**
Insignia 32" LED HDTV	$300	$191.69	$108.31
Arduino Uno	$20	$35.00	($15.00)
Leap Motion	$90	$88.98	$1.02
Speakers	$25	$24.00	$1.00
Intel Core i3-4340	$150	$159.99	($9.99)
Mini-ITX Case	$50	$49.99	$0.01
ASRock mini-ITX Motherboard	$100	$94.99	$5.01
ADATA 64GB SDD	$50	$69.99	($19.99)
G.SKILL 4GB DDR3 RAM	$50	$47.99	$2.01
Antec 380W PSU	$50	$44.99	$5.01
TP-Link N900 Wireless Adapter	$30	$42.92	($12.92)
Webcam	$80	$70.49	$9.51
Two-Way Mirror	$100	$68.91	$31.09
PCB Parts	$30	$69.84	($39.84)
Mirror Housing	$100	$52.27	$47.73
PCB Manufacturing	$100	$174.51	($74.51)
Document Printing	$100	$140	($40.00)
Total:	$1425	$1426.55	($1.55)

Table 8.2.1 – Original vs Final Budget

9.0 Project Summary and Conclusions

This documentation was about the smart mirror project. This stemmed from the need for better time management and productively along with the inspiration of new, developing technologies now available. The smart mirror idea was created to give instant access to information in a convenient and time-saving environment, the bathroom. All other aspects of the mirror's design developed from these ideas and inspirations.

The goals of the smart mirror were to aim to reduce time needed in a user's daily routine and provide a merger of user and technology that becomes an enhancement, not a new burden. The functionality must meet these descriptions in the design. The smart mirror did the thinking for the user with intelligent, commonly used applications. Apps like their calendar, music, news, Twitter, to-do lists, and weather will be available. The apps were unobtrusively displayed on the screen, hidden by the two-way mirror, as to look like a seamless experience. The user didn't even have to worry about turning on and off the system because the mirror will detect motion and do the work for them.

A good project can't be produced without proper research first. Similar projects and products were analyzed for similarities, improvements, and flaws. The group researched each important parts of the mirror system such as the gesture control, voice control, MCUs, and others. Once enough information was collected about specifications and prices, strategic components were selected to be part of the project from both a hardware and software perspective. The hardware components included the central PC components, the webcam, MCU, sensors, speakers, and Leap Motion Controller.

After research, the design phase was started. This included multiple subsystems which ranged in various difficulties and depth of design. One of the easier subsystems to design was the gesture control subsystem. Despite looking to be more difficult in the prototyping stage, the essential design elements use the Leap Motion API and their gesture class to setup which gestures will do what in the system. The other subsystems fell somewhere in the middle of easy and difficult with the temperature regulation subsystem and control software subsystem being the most intensive. The temperature system involved the knowledge of hardware design for the PCB and general temperature reduction methods. These had to be designed in a way that was time permitting, effective, and easily integrated. Other the other hand, the control software was composed of the system code organization as well as handling the UI and the GUI design. This involved knowledge of the available APIs, scheduling, and general intuitiveness.

The final stages of the project, and document, covered how the mirror was approached in terms of prototyping and testing. These were the focuses of Senior Design II. These were laid out to give order and direction for the group to stay organized. Overall, the group was optimistic and confident that the individual designs of the components would come together to deliver the promised smart mirror experience. The prototype that was delivered was well tested, contained minimal bugs, and was something that has that consumer product potential. The group continued to follow the milestones as well as the direction setup by this document to finally deliver the smart mirror as the best it can be.

Appendix A: Copyright Permissions

National Instruments

November 6, 2013

Chris Chiarella
University of Central Florida
c.chiarella7@gmail.com

Re: Permission to Use Certain Copyrighted Materials of National Instruments Corporation ("NI")

Dear Chris:

In response to your request to use certain copyrighted materials of NI, by this letter, NI grants you the limited right to reproduce Figures from the NI Developer Zone article entitled "Advantages of Solids State Drives (SSD) in PXI Systems" authored by National Instruments Corporation, April 1, 2013, located http://www.ni.com/white-paper/7482/en/ for inclusion in your senior design project document at the University of Central Florida.

1. You may use the NI Materials only for the purpose(s) described above (the "Purpose"). Please note that (unless expressly stated above in the Purpose) this is a one-time use right, and any additional uses by you will require the separate prior written authorization of NI.

2. Other than resizing the NI Materials as necessary for the Purpose, you may not revise, modify, or otherwise change the NI Materials in any manner without the prior written authorization of NI.

3. In no event may you use the NI Materials in a defamatory, libelous, or unlawful manner or in any manner that disparages NI.

4. You must retain all copyright and other proprietary notices of NI in the NI Materials. Without limiting the foregoing, you must include in the verification procedure, a copyright notice in prominent type and font indicating: (a) NI's copyright ownership of the NI Materials, and (b) that reproduction and redistribution of the NI Materials is courtesy of NI.

Please note that your use right is non-exclusive and non-transferable. NI reserves all rights and licenses not expressly granted you in this letter. **THE NI MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, AND NO WARRANTIES (EITHER EXPRESS OR IMPLIED) ARE MADE WITH RESPECT TO THE NI MATERIALS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, OR ANY OTHER WARRANTIES THAT MAY ARISE FROM USAGE OF TRADE OR COURSE OF DEALING. YOU AGREE AND ACKNOWLEDGE THAT NI SHALL NOT BE LIABLE FOR ANY CLAIMS RESULTING FROM OR IN CONNECTION WITH ITS USE OF THE NI MATERIALS, AND YOU AGREE TO INDEMNIFY AND HOLD NI HARMLESS FROM ANY CLAIMS AGAINST OR EXPENSES OF NI ARISING OUT OF OR IN CONNECTION WITH ITS USE OF THE NI MATERIALS.**

Sincerely,

Pete Smits
Sr. Intellectual Property Counsel
National Instruments Corporation

11500 N Mopac Expwy • Austin, TX 78759-3504 USA
Sales and Service: (512) 683-0100 • Other Calls: (512) 338-9119 • Fax: (512) 683-8411 • info@ni.com

Rtings

From: "Rtings.com" <3dtv@rtings.com>
Date: November 25, 2013 at 2:10:59 PM EST
To: "'Vishal Nagda'" <vishalnagda08@gmail.com>
Subject: RE: Image Permission

Yes, you can use it in your project document. Just list our website your sources/bibliography.

Cedric Demers
Rtings.com

From: Vishal Nagda [mailto:vishalnagda08@gmail.com]
Sent: Monday, November 25, 2013 1:57 PM
To: admin@rtings.com
Subject: Image Permission

Hello,

I am a student at the University of Central Florida working on my senior design project and would like to use your figure "Optimal viewing distance by the size of the television and resolution" from this page http://www.rtings.com/info/television-size-to-distance-relationship in my design document. We are required to write and design a document for our project and it would be for educational purposes only. Would it be possible to obtain permission to use this figure?

Thank you.

Vishal Nagda
vishalnagda08@gmail.com
407-766-6586

Leap Motion

Content

All text, graphics, user interfaces, visual interfaces, photographs, trademarks, logos, sounds, music, artwork and computer code (collectively, "Content"), including but not limited to the design, structure, selection, coordination, expression, "look and feel" and arrangement of the Content, contained on the Site is owned, controlled or licensed by or to Leap Motion, and is protected by trade dress, copyright, trademark laws, and various other intellectual property rights and unfair competition laws.

You may use information on Leap Motion products and services (such as data sheets, knowledge base articles, and similar materials) purposely made available by Leap Motion for downloading from the Site, provided that you (1) not remove any proprietary notice language in all copies of such documents, (2) use the information only for your personal, non-commercial informational purpose and do not copy or post such information on any networked computer or broadcast it in any media, (3) make no modifications to the information, and (4) not make any additional representations or warranties relating to the information.

MIT Open Courseware

MIT Interpretation of "Non-commercial"

Non-commercial use means that users may not sell, profit from, or commercialize OCW materials or works derived from them. The guidelines below are intended to help users determine whether or not their use of OCW materials would be permitted by MIT under the "non-commercial" restriction. Note that there are additional requirements (attribution and share alike) spelled out in our license.

1. **Commercialization is prohibited.** Users may *not* directly sell or profit from OCW materials or from works derived from OCW materials.

   Example: A commercial education or training business may not offer courses based on OCW materials if students pay a fee for those courses and the business intends to profit as a result.
2. **Determination of commercial vs. non-commercial purpose is based on the use, *not* the user.** Materials may be used by individuals, institutions, governments, corporations, or other business whether for-profit or non-profit so long as the use itself is not a commercialization of the materials or a use that is directly intended to generate sales or profit.

   Example: A corporation may use OCW materials for internal professional development and training purposes.
3. **Incidental charges to recover reasonable reproduction costs may be permitted.** Recovery of nominal actual costs for copying small amounts (under 1000 copies) of OCW content on paper or CDs is allowed for educational purposes so long as there is no profit motive and so long as the intended use of the copies is in compliance with all license terms. Students must be informed that the materials are freely available on the OCW Web site and that their purchase of copied materials is optional.

   Example: An institution in a remote area has limited Internet access and limited network infrastructure on campus, and a professor offers to create CDs of OCW materials relevant to her course. The professor may recover the costs of creating the CDs.

Questex Media

2. CONTENT; COPYRIGHTS; TRADEMARKS

a) All of the information, content, services and software displayed on, transmitted through, or used in connection with the Site, including advertising, directories, guide text, photographs, images, illustrations, audio clips, video, html, source and object code, software, data, and all other matters related to the Site, and including the sele aforementioned and the "look and feel" of the Site (collectively, the "Content"), are: Copyright © 2009 Questex Media Group, Inc., 275 Grove Street, Suite 2-130, New content providers, and are protected by U.S. and international copyright laws. All rights reserved.
b) All Content is protected by copyright, and owned or controlled by QMG or the party credited as the provider of the Content. You shall abide by all additional copyri restrictions contained in any Content accessed through the Site. If you have any questions about whether your Content is protected by copyright and the steps to tak copyright in your Content you should consult with an attorney. To the extent that copyright protection exists in your Content, you will retain ownership on such copyrig our site, subject to certain licenses granted to us and/or parties who wish to use your Content, as more particularly described herein.
c) The Content made available on the Site or in connection with the Services is intended for your personal, noncommercial use. You may use the Content online and commercial use, and you may download or print a single copy of any portion of the Content for your personal, non-commercial use, provided you do not remove any notice contained in such Content. Other than as expressly provided in this Agreement, no other use is permitted.
d) You may not modify, copy, frame, rearrange, distribute or redistribute, reproduce, sell, publish, transmit, display or otherwise use any of the Content or any part c advertising, whether or not for payment or other consideration, without prior written permission from QMG.
e) In exchange for the access provided to the Content, you agree that you will not remove, obstruct, modify or otherwise interfere with the delivery or display of adver
f) The Content includes logotypes, trademarks and service marks owned by QMG and by other information providers and third parties, none of which may be used in in advance and in writing, by QMG.
g) Requests to use the Content for any purpose other than as permitted in this Section 2 should be directed to the appropriate QMG publisher; if you are unsure, plea

Texas Instruments

Use Restrictions

The Materials contained on this site are protected by copyright laws, international copyright treaties, and other intellectual property laws and treaties. Except as stated herein, these Materials may not be reproduced, modified, displayed or distributed in any form or by any means without TI's prior written consent.

TI grants permission to download, reproduce, display and distribute the Materials posted on this site solely for informational and non-commercial or personal use, provided that you do not modify such Materials and provided further that you retain all copyright and proprietary notices as they appear in such Materials. TI further grants to educational institutions (specifically K-12, universities and community colleges) permission to download, reproduce, display and distribute the Materials posted on this site solely for use in the classroom, provided that such institutions identify TI as the source of the Materials and include the following credit line: "Courtesy of Texas Instruments". Unauthorized use of any of these Materials is expressly prohibited by law, and may result in civil and criminal penalties. This permission terminates if you breach any of these terms and conditions. Upon termination you agree to destroy any Materials downloaded from this site.

Appendix B: References

[1] http://blog.litstudios.com/index.php?/archives/14-Interactive-Mirror.html
[2] http://www.tofsen.se/articles/30/revolutionizing-your-bathroom-experience
[3] http://www.extremetech.com/computing/94751-the-new-york-times-magic-mirror-will-bring-shopping-to-the-bathroom
[4] http://www.cybertecturemirror.com/main.php?id=home
[5] http://www.engadget.com/2010/10/13/cybertecture-mirror-reflects-our-fantasies-looks-set-to-become/
[6] http://www.theverge.com/2012/5/10/3013168/seraku-android-mirror-prototype-hands-on
[7] http://www.youtube.com/watch?v=uF0NSUmxFYA
[8] http://blog.seattlepi.com/microsoft/2010/11/05/the-guts-of-microsofts-kinect-sensor/
[9] http://www.cpelectronics.co.uk/
[10] http://www.nintendoworldreport.com/news/11557
[11] http://www.gamasutra.com/php-bin/news_index.php?story=24456
[12] http://www.newegg.com/Product/Product.aspx?Item=9SIA25V0YW6553
[13] http://www.lionprecision.com/tech-library/technotes/cap-0020-sensor-theory.html
[14] http://www.ni.com/white-paper/7482/en/
[15] http://www-03.ibm.com/ibm/history/exhibits/storage/storage_350.html
[16] http://www.newegg.com/Product/Product.aspx?Item=N82E16822178265
[17] http://www.newegg.com/Product/Product.aspx?Item=N82E16820248017
[18] https://www.leapmotion.com/product
[19] http://hometheater.about.com/od/hometheatervideobasics/qt/720p-Vs-1080p.htm
[20] http://www.hgtv.com/home-improvement/light-bulbs-know-the-different-types/index.html
[21] http://www.lutron.com/en-US/Education-training/Pages/LCE/DimmingCFLsandLEDs.aspx
[22] http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/temperature-measurement/platinum-rtd-sensors/microcontroller
[23] http://www.ti.com/lit/an/snoa663b/snoa663b.pdf
[24] http://www.cryocon.com/S900/S900ds.pdf
[25]http://www2.latech.edu/~dehall/LWTL/ENGR121/notes/11_RTD_overview_student.pdf
[26] http://www.princeton.edu/~cavalab/tutorials/public/Thermocouples.pdf
[27] http://download.tigal.com/veear/SmartVR_User_Manual.pdf
[28] http://ww1.microchip.com/downloads/en/DeviceDoc/21895d.pdf
[29] http://www.emartee.com/product/42120/
[30] http://www.13thmonkey.org/documentation/hardware/chips/80-0116.pdf
[31] http://www.sensoryinc.com/support/docs/80-0184-I.pdf
[32] http://www.tigal.com/product/1900
[33] http://www.tigal.com/product/1770

[34] http://www.wwdmag.com/water/seven-basic-types-temperature-sensors
[35] http://www.engineeringtoolbox.com/temperature-sensors-d_448.html
[36]http://www.maximumpc.com/article/features/two_gaming_technologies_explained_white_paper_round-?page=0,0
[37]http://www.newegg.com/Product/Product.aspx?Item=26-104-635&SortField=0&SummaryType=0&Pagesize=10&PurchaseMark=&SelectedRating=-1&VideoOnlyMark=False&VendorMark=&IsFeedbackTab=true&Page=3#scrollFullInfo
[38]http://www.visualstudio.com/
[39]http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx
[40] http://www.toodledo.com/index.php
[41] http://arduino.cc/en/Main/ArduinoBoardUno
[42] http://en.wikipedia.org/wiki/Noise-canceling_microphone
[43] http://www.veear.eu/products/easyvr/
[44] http://www.wwdmag.com/water/seven-basic-types-temperature-sensors
[45] http://www.engineeringtoolbox.com/temperature-sensors-d_448.html
[46] http://www.ti.com/lit/an/snia009/snia009.pdf
[47] http://www.ecnmag.com/articles/2013/04/ntc-thermistors-versus-voltage-output-ic-temp-sensors
[48] http://www.ti.com/lit/ds/symlink/lm50-q1.pdf
[49] http://automationwiki.com/index.php?title=Semiconductor_Temperature_Sensors
[50] http://www.digikey.com/product-search/en/sensors-transducers/humidity-moisture/1966708
[51] http://www.sensorsmag.com/sensors/humidity-moisture/choosing-a-humidity-sensor-a-review-three-technologies-840
[52] http://www.rtings.com/info/television-size-to-distance-relationship
[53] http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/overview.page
[54] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-071j-introduction-to-electronics-signals-and-measurement-spring-2006/lecture-notes/24_op_amps3.pdf
[55] http://www.ist-usadivision.com/sensors/humidity/
[56] http://en.wikipedia.org/wiki/ICalendar
[57] https://dev.twitter.com/docs/api/1.1
[58] http://msdn.microsoft.com/en-us/library/hh920511(v=vs.85).aspx
[59] http://www.gskill.com/en/series/desktop-memory
[60] http://www.intel.com/content/www/us/en/processors/core/core-i3-processor.html
[61] http://www.asrock.com/general/products.asp
[62] http://www.antec.com/?page=cookieSet&url=/product.php
[63] http://www.tp-link.us/products/
[64] http://www.adata-group.com/product_all.html
[65] http://www.adafruit.com/products/357#Tutorials
[66] http://www.aaroncake.net/electronics/solder.htm
[67] http://www.electroschematics.com/6244/humidity-control-switch/
[68] http://www.ti.com/lit/ds/symlink/lm35.pdf

[69]http://www.analog.com/static/imported-files/data_sheets/TMP35_36_37.pdf
[70] http://www.analog.com/static/imported-files/data_sheets/TMP35_36_37.pdf
[71] https://developer.att.com/apis/speech/docs
[72] http://api.toodledo.com/3/index.php
[73] http://stevenhickson.blogspot.com/2013/05/using-google-voice-c-api.html
[74] http://www.nuance.com/for-developers/dragon/index.htm
[75] http://cmusphinx.sourceforge.net/