

# Near Field Communication Operated Programmable Robotic Toy

Renso Pasara, Noel Abdon, Anthony Wu,  
Gabriel Martin

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

**Abstract** — This paper outlines the design and functionality of a robot that serves to expose users to foundational programming concepts. The core project revolves around a four-wheel robotic car with a dynamic turtle motif, complete with a shell that serves interactive functionality. Its distinguishing feature is the utilization of fifteen Near Field Communication (NFC) sensors, arrayed into slots on the car's shell, allowing user inputs via command blocks which are compiled and executed using buttons on the shell. The robot exhibits high maneuverability using individually controlled mecanum wheels, supported by fifteen feedback LEDs, piezoelectric buzzers, and four ultrasonic sensors.

**Index Terms** — Near field communication, passive RFID tags

## I. INTRODUCTION

We introduce a four-wheeled, turtle-themed toy robot which serves as not only a playful gadget but an educational tool to simplify the complex concepts of programming for children. Our robot utilizes fifteen coding blocks, each embedded with a RFID readable tag, to issue commands in a user-friendly, immersive manner, allowing seamlessly interchanging and matching of various blocks. We have sought to incorporate a practical and engaging approach to introduce programming to children unlike traditional methods that rely heavily on tablets or computer programs. The device primarily relies on NXP's PN532 NFC (Near Field Communication) reader modules, with SPI, in order to read and interface with the blocks themselves, arranged smartly on the chassis of the robot. Our chassis is composed of a metal framework with a 3D printed shell and pieces, as well as the blocks themselves. Each block is written with a string of data that pertains to commands, and when read by the robot, will be executed sequentially by the robot. To match the fifteen readers, fifteen LEDs and four buzzers provide visual and auditory feedback to the various commands and states of the robot itself. Four ultrasonic sensors around

the cardinal directions of the robot provide a precaution against obstacles during operations, as well as another method to engage with the device. An ATmega2560 provides the sufficient number of pins and functionality to handle the number of readers and hardware required.

## II. SYSTEM COMPONENTS

### A. Microcontroller (MCU)

The Arduino Mega 2560 was selected as the microcontroller for our project due to its large number of digital pins and widespread use in beginner-friendly robotics. Its simplicity, extensive documentation, and availability of components made it a comfortable choice for the team, facilitating prototyping and testing while meeting the necessary features for our project. The mega boasts a total of 54 input/output pins with 15 of these pins being able to be used for Pulse-width modulation should a component we are adding require it. The 54 pins allows us to comfortably accommodate the large amount of peripherals we have to connect for our project. This prevents the need of additional circuitry such as an SPI extender that would be needed otherwise to add pins to our microcontroller. [1]

### B. RFID Module

The PN532 NFC/RFID Module, by NXP, was selected for its versatility and robust performance in RFID technology. NXP Semiconductors is a leading supplier in identification ICs and tagging. They provide a number of tags and support various formatting, which our design incorporates.

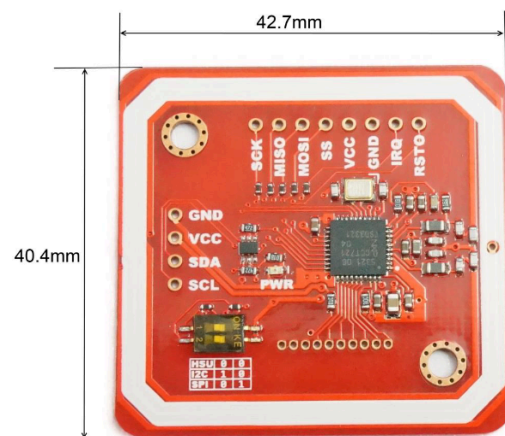


Fig 1. Full image of an NXP PN532 NFC reader, showing it's layout and size [3]

Supporting both RFID and NFC, and operating at 13.56

MHz, this module offers compatibility with various RFID card formats and supports multiple protocols, including I2C, SPI, and HSU (High Speed UART). A frequency of 12.56 MHz makes this a high frequency system, making it well capable of passing through materials. Its reliability and ease of integration make it an ideal choice for our project's diverse application needs. The antenna built into the reader has a 5cm~7cm operational range for communication, ideal for reading through thin plastic, but not far enough as to possibly interfere with nearby readers, as our design calls for them to be arranged on a surface. As well as its default operating voltage being 5 volts allows us to implement the device without the need of an additional voltage regulator for it. This module in particular, on top of being versatile and cheaper, is also heavily documented with a reliable open source library available for the Arduino IDE. This is another significant factor in our decision to utilize the module. Our previous choice of RFID was the MFRC522 module, which was cheaper, and with similar library support. However, in application, we were unable to reliably operate the reader with two or more modules connected at once. Further testing and research with necessary tools (oscilloscopes, SPI signal readers) led us to conclude that the MFRC522 is unsuitable for our application due to interference with itself (among identical MFRC522 modules) when connected using SPI. I2C was similarly unusable due to lack of possible configuration of the addresses. Additionally, it required a voltage regulator of 3.3 volts. Due to this it was eliminated from consideration after further testing allowing for us to reduce the amount of circuitry required to run our designs. [1][2]

### C. DC Gear Motor

After careful consideration, we chose the Greartisan DC Gear Motor for its optimal speed and stall torque. With a speed of 60 RPM and a stall torque of 6.5 kg-cm per motor, this motor strikes a balance between speed and torque, aligning perfectly with our project requirements. Its low current consumption ensures efficient power utilization, maximizing battery life during operation. The motor also operates at a voltage level of 12 volts which perfectly aligns with the battery we chose to power our project as it outputs 12 volts. The integration with our other components was very simple, as all the motors needed to operate were cables to supply them with power. All these features made it the reliable and sufficient choice for our project without being complete overkill.

### D. Motor Driver

The HiLetgo L298N motor driver was selected for its

compatibility with our 12V DC motors, bidirectional control capability, and robust design. Its dual H-Bridge configuration allows for efficient control of two motors simultaneously, while built-in protective features such as large-capacity capacitors and diodes ensure safe operation. Additionally, the integrated heat sink enhances thermal dissipation, contributing to the motor driver's reliability during operation. The heat sink should help and prevent thermal runaway on the motor driver chips which are known to overheat and could potentially cause an accident without the heatsink. Overall, the HiLetgo L298N motor driver meets our project requirements for motor control and safety.

### E. Battery

Component	Number of Components	Voltage (V)	Current (mA)
Motors	4	12	240
Motor Drivers	2	5-36	74
Arduino	1	5	60
RFID	15	3.3	450
Ultrasonic Sensor	4	5	60
Speaker	2	3	40

Fig 2. Total Current Consumption breakdown

The Talentcell Rechargeable Battery stood out as the optimal power source for our project. With a 12-volt output and a capacity of 6000 mAh, it perfectly matches our motor's voltage requirement while providing ample capacity for extended operation. Despite its weight and price, the Talentcell battery's large capacity ensures future expandability without compromising battery life, making it an ideal choice for our project. As well as the additional feature of a 5 volt usb port which can provide power to our MCU through the usb port allowing for us to save on needing to buy additional wires and circuitry to power the MCU.

Based on our total current consumption calculations the battery should last for around 5 hours with the current estimated amount of sensors and peripherals. This is perfect for our application as it is expected to be used with children for hours at a time and will ensure the children have access to the device throughout the day at a daycare or school setting application.

The main reason this battery was chosen was for its reliability and protective circuit. This battery has equipped a robust and thorough protective circuit that ensured we did not cause an electrical fire while constructing the device or while the device is in use by the children. The protective circuit would . It was recommended to us for this exact reason.

### *F. Ultrasonic Sensor*

For object detection, we selected the HC-SR04 ultrasonic sensor. Its affordability, reliability, and proven performance in collision detection made it the optimal choice for our project. With a range of 400 cm and excellent accuracy, coupled with our team's familiarity with its usage, the HC-SR04 sensor ensures reliable collision detection in various scenarios. The sensor was selected mainly for the familiarity our entire group has with the sensor and does the job required of it effortlessly as we only need it to detect objects within a very close range of the device.

### *G. LED Driver*

This shift register, the 74HC595, features serial to parallel conversion capabilities with the potential for controlling brightness through external components. While it lacks an integrated constant current sink inherent to designs like the TLC5940 which we had considered, it can be used in conjunction with external resistors to manage current, thus contributing to the efficient performance and durability of LEDs. By utilizing external circuitry for dimming and color control, the 74HC595 offers a versatile approach to LED control. This setup requires careful management to protect LEDs from voltage and current variations, ensuring a stable operation conducive to the longevity of LED applications. To achieve this control we implemented a low value resistor to help protect the LEDs we use for our project. The shift register works perfectly as we just need an indication with the LED to show that the RFIDs are operating and reading when they are supposed to.

### *H. Voltage Regulator*

For voltage regulation, we chose the TPS564257DRLR for its efficiency and stable voltage output. The TPS563300DRLR offers a stable 5 volt output with an amperage limit of 2 Amps. This regulator ensures stable voltage supply to our components, maximizing efficiency and reliability in our project. The voltage regulator is essential to the design as it ensures the slightly fluctuating levels of voltage output from our battery won't damage our device over time as the battery loses charge.

### *I. Piezoelectric Buzzard*

For our design, our group wanted to include sound feedback to users when the RFIDs were being read successfully and when nothing was read at all. This helped a lot with programming and will be fun with the kids who

use it as they have another sensory element to interact with. A stretch goal for our project was to add blocks that can play music for the kids when placed on the device. The piezoelectric buzzer, leveraging the inherent properties of piezoelectric materials, excels in converting electrical signals into audible sounds through mechanical vibrations. Unlike systems with built-in amplification or modulation capabilities, such as certain advanced audio output devices, the piezoelectric buzzer operates on the direct application of voltage to elicit sound. This simplicity, however, doesn't detract from its effectiveness or versatility. By integrating it with external circuits for frequency modulation, one can achieve a range of tones, from simple beeps to more complex sequences of sound, making it adaptable for various signaling or user interface feedback applications.

## III. HARDWARE DETAILS

The hardware design of the project encompasses two main components: the supporting structure and the technology integration. First, we discuss the design of the robot car itself, including the chassis, motor and wheel mounts, and aesthetic choices. Then, we delve into how inputs are received and processed by the Arduino on the PCB.

The design of the robotic car prioritizes visual appeal, safety, and durability, especially considering its intended audience of children. The chassis, constructed from aluminum parts, ensures sturdiness, while exterior components are made of plastic with filed edges for safety and comfortability. The use of bright green PLA plastic enhances the car's visual appeal, aligning with the turtle-themed design.

The chassis utilizes Actobotics aluminum robotics construction components, featuring U-Channels and 90-degree connectors. These parts provide flexibility in mounting positions and wiring management. Plates attached to the underside of the car support wiring and provide a centralized location for the mounting of the PCBs and additional components.

Mecanum wheels were chosen for omnidirectional movement flexibility. The wheels are positioned to allow the car to move in any direction by activating specific motors strategically. A plastic cover resembling turtle legs encases the wheels, facilitating maintenance and addressing any safety concerns.

Four motors, configured for individual control, are placed around the vehicle. This design was chosen as opposed to an axle approach, due to its simplicity, despite needing double the amount of motors. Each motor, and therefore each wheel, can turn independently, enabling

precise movement and maneuverability. These movement options include all four cardinal and intercardinal directions as well as rotation, both clockwise and counterclockwise.

To receive these movement commands from the user, various blocks will be available, each containing its own movement instruction. These blocks are designed to be placed on the top of the robot in one of fifteen slots for the user to choose from. The blocks were printed with an RFID tag embedded just below the surface to allow the PN532 module to consistently and accurately read the data it contains. These blocks are light and comfortable to handle making them perfect for use in a classroom setting. The figure below depicts a cross section of the coding block, showcasing the placement of the RFID tag to minimize physical interference that the PLA material may cause. Feedback will be provided to the user both audibly and visually through the use of a buzzer with two distinct sounds, indicating a successful or unsuccessful read, and LEDs that will power on or off showing that same status.

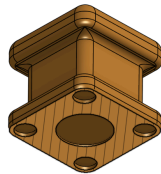


Fig 3. CAD model of a subsection of the coding block, showcasing the placement of the RFID tag just below the surface of the PLA.

The overall hardware system consists of the driving subsystem and the sensor subsystem, both essential for the functionality of the robotic car.

#### A. Driving Subsystem

The driving subsystem receives inputs from the RFID/sensor subsystem, processes instructions, and powers the motors accordingly.

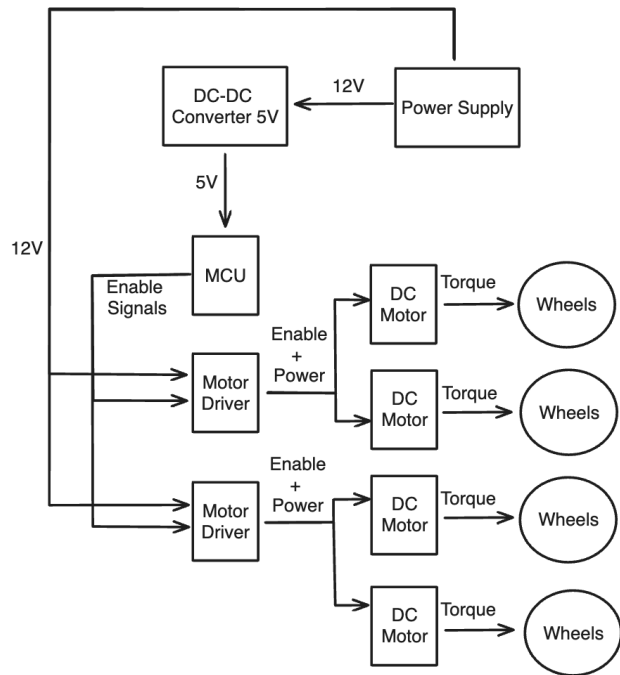


Fig 4. Block diagram of the driving subsystem, showing how information and power flows in the system.

The motor drivers operate using input signals in order to determine the direction of rotation for each individual wheel. As such, it requires at minimum eight digital IO pins to operate (Two pins per wheel) in order to determine direction and activation. It consists of the following components:

- Motor Drivers: L298N motor drivers interface with the Arduino to control motor movement effectively. Each motor driver handles two motors, employing a dual H-bridge design for bidirectional control.
- Power System: A 12V power supply powers the entire system. DC-DC converters regulate voltage levels, supplying 5V to the MCU and 12V to the motor drivers.
- MCU: The Arduino receives instructions from the RFID/sensor subsystem and sends signals to the motor drivers to execute motor commands.

#### B. Sensor Subsystem

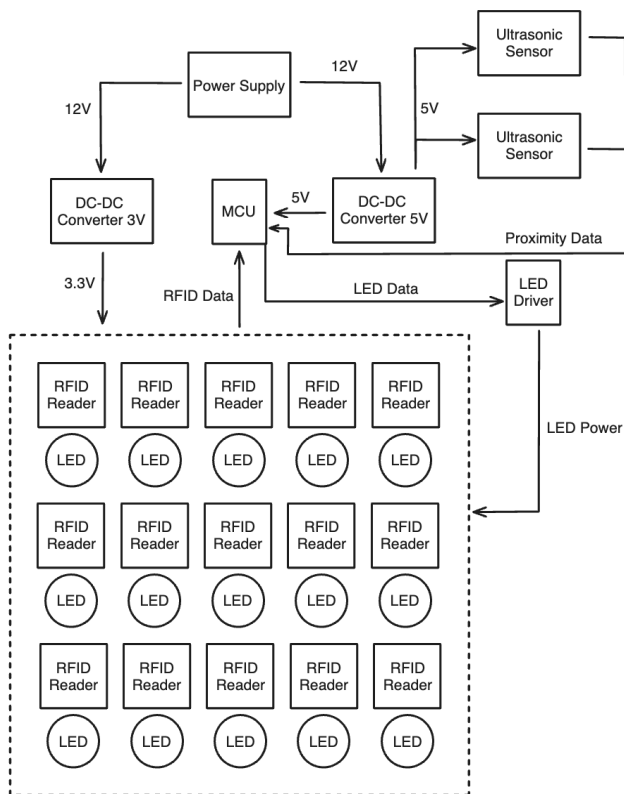


Fig 5. Block diagram of the sensor subsystem, showing how information and power flows in the system.

The sensor subsystem gathers input from user-coded blocks, RFID reader, and ultrasonic sensors, providing feedback before executing motor commands. It comprises the following components:

- RFID Sensors: An array of RFID sensors interacts with RFID tags on coding blocks, enabling the Arduino to decipher instructions.
- Ultrasonic Sensors: Placed at the front, sides, and back of the car, ultrasonic sensors detect objects for collision prevention.
- LED Communication: LED indicators provide feedback on block recognition. An LED driver controls individual LEDs, indicating block status and system errors.

These subsystems work in tandem, with the driving system translating user inputs into motor commands for the robot car's movement, ensuring a seamless and interactive user experience.

#### IV. SOFTWARE DETAILS

All software design is done on the Arduino IDE due to the lack of graphical user interface. There is no user

interaction outside of the mechanical pieces of the robot, so interfacing with the user was not necessary. This also aligns with our goals of keeping the robot as simple to learn and use, seeing as our target audience is children as young as 5 years of age. Our goals when designing the robot's programming focused mainly on runtime, feedback, and modularity. In summary, the robot needs to respond to the user's actions as quickly and as clearly as possible through lights and sounds. Lastly, due to the nature of our project, modularity in the blocks was important. We want to allow the user to place any combination of blocks on the robot and have it follow each command in order. To do this, each block should successfully execute its command no matter what other blocks have been placed on the other slots.

##### A. Three State Design

To allow for easy testing and clear functionality, the group decided on implementing three different states the robot can be in: Standby, Learning, and Executing. Separating the robot's functions the developers know where the error is whenever the robot malfunctions as well as let the user know what the robot is supposed to be doing at any given moment.

In the standby mode, the robot is awaiting for user input. In this mode, the robot might be waiting to enter the learning or executing mode, depending on if the user has already loaded a set of commands in its memory. During standby mode, the robot is stationary and it is the mode the robot enters when it is first powered on. This is the simplest to implement in programming as the project is awaiting instructions.

Most commonly, the robot will enter the Learning state after the user pushes the appropriate button on the project. In this mode, the RFID sensors are activated one by one and attempt a read on which blocks are placed. After the activation of an RFID, an LED next to the respective slot will show the result of a read, with a lit LED signifying a successful read and a shut off LED signifying some kind of error. Also during this process, the software is storing commands into the command queue in order in which the user has placed the blocks. How this queue works is explained in subsection B. Once all RFIDs have attempted a read, the robot re-enters standby mode to await further commands. The goal with this state is to introduce the user to the concept of compiling and debugging actual computer code in software such as an IDE.

The last state is the Executing state, where the user sees the robot perform the functions they have coded. The robot enters this state from the standby state when they press the execute button. Referring to the command queue,

the robot knows what order it needs to perform each function. During this state, the robot's onboard ultrasonic sensors are activated and alert the robot of its surroundings. If during a command the robot is to approach an obstacle and crash, the ultrasonic sensors will abort the current command and move on to the next. Also, the user has the option of aborting all commands in case of emergency by pressing the execute button once again. Once all commands are executed or aborted, the robot re-enters standby state, with the same code still stored in memory. This allows the user to retry executing the same code. This mode is designed to most closely resemble executing actual code.

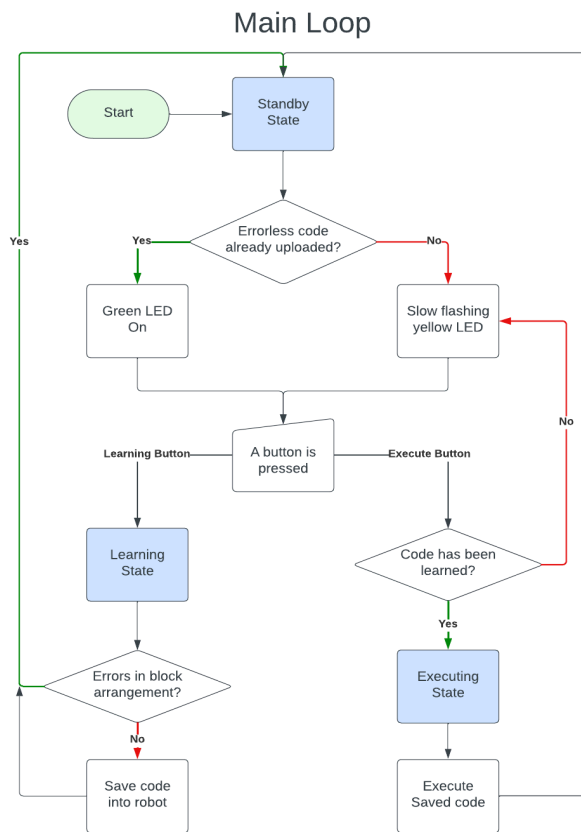


Fig 6. State Diagram Flowchart

### B. Memory Structures

The microcontroller stores all data necessary for execution in two queues. The first queue is filled during the Learning state, and it is called the “command queue”. This queue is a FIFO queue (first in, first out) to maintain the order of the user’s commands intact. As the robot performs the commands in the queue, it dequeues that command from the command queue and enqueues it onto the second queue, what is called the “executed queue”. This secondary queue is implemented in the case of a

Repeat or Repeat All block. If either of these two repeat blocks are found in the command queue, then the robot will refer to the executed queue to know which commands are to be repeated and in what order. An example of this process can be seen in the figure below.

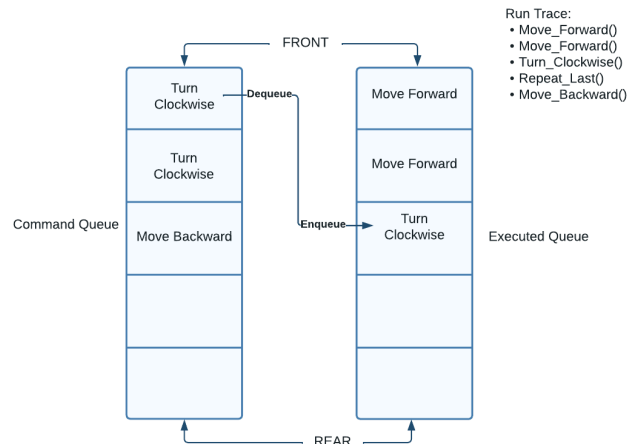


Fig 7. Command and Executed Queue visualization with example

The state of these commands are constantly changing during the executing and learning state. Specifically, the command queue is cleared at the start of the learning state and all of the executed queue is copied into the command queue at the end of the execution state. After this, the executed state is cleared.

### C. Functions

The code is in C++ for its compatibility with the Arduino IDE, where all the programming has happened. The code for each block is compartmentalized into individual functions so they are easily accessible from anywhere in the code. The commands include movements such as MoveForward() and TurnClockwise() as well the repeat blocks.

The code is also equipped with several helper functions that help the management of the command queues as well as the control of the two subsystems. The control of the shift register accessing the LEDs is also its own function. Additionally, the executing and learning states are segmented into their own functions for ease of debugging. These are where most errors during development would appear.

### D. SPI (Serial Peripheral Interface)

SPI, or Serial Peripheral Interface, is a synchronous serial communication protocol created by Motorola. This interface is employed primarily for short-distance

communication and in microcontroller-based systems. It uses a master-slave configuration and operates on a full-duplex mode, enabling simultaneous data transmission from master devices to slaves and vice versa.

Technically, SPI consists of a clock signal (SCLK), a master data output (MOSI), a master data input (MISO), and slave select (SS) lines. The clock signal helps synchronize data transmission between the master and the slave devices. The MOSI allows the master to send data to the slave, while the MISO lets the slave send data to the master. The SS lines determine to which slave device the information should be sent.

The SPI protocol's simplicity, high-speed operation, and flexibility make it a popular choice for applications involving data streaming, sensor data reading, and real-time requirements, among others. Furthermore, the actual hardware and implementation aspects are straightforward. This relative simplicity makes it ideal for applications not requiring complex configuration or high levels of error control.

We have fifteen NXP PN532 modules, equipped to read tags from the coding blocks, connected to one another using this interface. By implementing SPI, we manage efficient communication between our modules and the microcontroller, which acts as the master.

### E. PN532 Programming

To program the functionality for the RFIDs, the code references libraries specifically made for the PN532. The library is compatible with several communication protocols, but only the SPI features were needed for the purposes of the project.

Tying each RFID sensor with a slave pin on the microcontroller, the board is able to control which reader to receive input from at any given time. Even though the SPI communication protocol requires more digital pins on the microcontroller, it is superior in high speed and low power situations. Using a loop that activates one RFID at a time, each tag's data is stored in the command queue.

The type of tags being used were MIFARE Classic and they use NFC data exchange format (NDEF) to store messages [4]. Using this, each block containing a MIFARE Classic tag is coded to have an NDEF message stored in its memory. Within the NDEF message, several NDEF records are stored. One NDEF record is used to store a command as an 8-bit array in a payload variable. This record payload is used to identify the type of block the tag is in, such as a "Move Forward" block or a "Repeat" block.

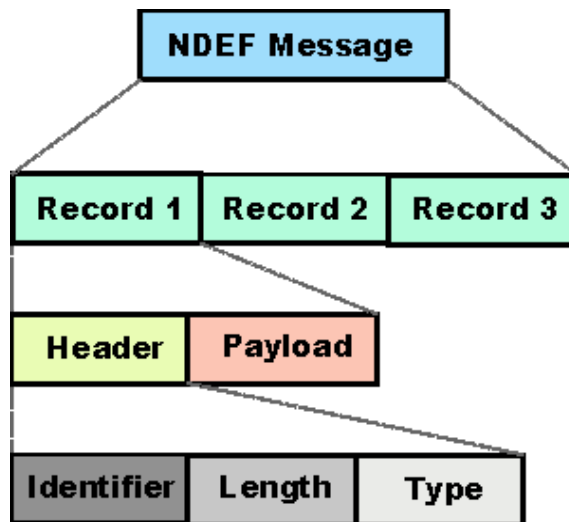


Fig 8. NDEF Message Structure [4]

The tags themselves are also manufactured with a unique code called a UID, but these are omitted due to the lack of modularity this would cause. The goal is to have many blocks of the same type, so a unique code for each tag is not as useful.

## V. BOARD DESIGN

### A. Overall Design

The final assembly and integration of the electrical components for our project will be facilitated through the creation of a custom Printed Circuit Board (PCB). After evaluating the option of using a standard, pre-made PCB. Our team decided to design and order a custom PCB for this project. This decision was reached following thorough discussions about the costs and necessary equipment, with further details on these components outlined in earlier sections of this document.

The overall board design was something very important to our project. The PCB shape was planned to be part of the structure of the device. This decision was made in order to eliminate the amount of hanging wires and possible points of failure from the rough interactions the device could potentially have with the children using it. Our device is split into two PCBs one for the MCU and PCU of our device and the other to connect and hold all of our sensors and LEDs.

The primary board is the brains of the operation as it holds our ATMEGA2560 and our voltage regulator and accompanying circuitry to ensure it can interface with our sensors. This board also has our Piezoelectric buzzard that used to make noise to either confirm or indicate failure of

a read of the RFID and an added stretch goal of adding noise blocks to our repertoire of block commands. As well as the pins broken out to ensure we can connect to our second board and potential devices we decide to add to the device in the future.

The second board is integral to the game-like aspect of our device as it would help space the RFID's in a grid to make the placement of the blocks on top of them more natural and intuitive for the kids. This pcb does a large portion of the wire routing to minimize the chances of a kid unplugging a sensor on accident through rough play.

### *B. PCB Design Software*

The PCB software our team chose for this project is Autodesk EAGLE. The main reason my team chose this as our PCB software was due to the familiarity everyone in our group has with EAGLE because of the information and experience we gained in our Junior Design course.

Autodesk Eagle, formerly known as CadSoft Eagle, is a versatile and robust software used in electronic design automation and the creation of PCBs. It's one of the more popular options when it comes to PCB designing in the electronic community for a multitude of reasons such as Large component library, lots of online resources, compatibility with other softwares, and longevity as a company.

Eagle has an excellent library of components already pre-installed in the software with an easy to use User interface that allows for a quick and seamless adding of parts. If the part you need is not listed on the pre-installed libraries, EAGLEs widespread use and large following has created multiple resources online where you can find the files you need to add the part you need for your schematic. This was another major factor when deciding what software to use as we can get assistance all over the world wide web thanks to EAGLES popularity.

### *C. PCB Manufacturing*

JLCPCB was the manufacturer of choice for our project. We chose this company due to its known reliability, speed, and price of manufacturing. JLCPCB, or Jiangsu New Pengfei Electronic Technology Co., Ltd., is a prominent player in the field of printed circuit board (PCB) manufacturing. Founded in 2006, JLCPCB has rapidly grown to become one of the largest PCB manufacturers globally, known for providing high-quality and affordable PCB fabrication services. The company is

headquartered in Shenzhen, China, a city renowned for its thriving electronics industry.

JLCPCB has revolutionized the PCB manufacturing process by leveraging advanced technologies and automated production systems. They offer a wide range of PCB services, including prototype fabrication, low-volume production, and assembly services. The company has established itself as a go-to solution for engineers, hobbyists, and businesses seeking reliable and cost-effective PCB solutions. This made it ideal for our group who is testing a potential product and can order a low volume for an affordable price. The main downside to this manufacturer is the distance. Due to their headquarters being located in China it raises the cost quite a bit for shipping but this cost is out weighed with the pros of low volume, quick, and efficient production.

## VI. CONCLUSION

The system, when all components work in tandem, create the perfect aid for coding and programming education. It promotes an environment of play and encourages the user to experiment with the robot's features to accomplish fun goals. In this process, they can learn coding fundamentals with a low cost of entry. No screens, long instructions, or strong reading skills necessary.

## ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Arthur Weeks; University of Central Florida.

## REFERENCES

- [1] Microchip Technology Incorporated, ATmega640/V-1280/V-1281/V-2560/V-2561/V Datasheet <https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf>
- [2] NXP Semiconductors, "UM0701-02 PN532 User Manual". <https://www.nxp.com/docs/en/user-guide/141520.pdf>
- [3] ELECHOUSE, "PN532 NFC RFID User Guide". [https://www.elechouse.com/elechouse/images/product/PN532\\_module\\_V3/PN532\\_%20Manual\\_V3.pdf](https://www.elechouse.com/elechouse/images/product/PN532_module_V3/PN532_%20Manual_V3.pdf)
- [4] Hou, Robert. "Introducing NDEF - NFC Data Exchange Format." Dialer Smart Card and RFID Label/Tag Shop, [www.gialer.com/blogs/news-and-blogs/introducing-n-def-nfc-data-exchange-format](http://www.gialer.com/blogs/news-and-blogs/introducing-n-def-nfc-data-exchange-format)