RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

Abstract—This report describes how RFIDJs designed and built a modernized jukebox for their senior design project. The purpose of this project was to showcase our engineering skills that we have acquired within these four years at Syracuse University. This project is important to RFIDJs and for the class since it demonstrates the skills we have learned. Between these two semesters we learned and experienced what it would be like working in the real world on a team project. In addition, we gained new skills when working with hardware and software components we had never used before. It not only showcased our skills but it demonstrated that RFIDJs can work as a team and complete a project. Additionally, this paper outlines in detail how the software and hardware components work together to create a working jukebox. It describes the actions RFIDJs took to create this project as well as the results. Furthermore, it describes what RFIDJs presented at the end of semester.

I . INTRODUCTION

Between the 1940s and the 1960s, jukeboxes were very popular to the public. They were set up in many different locations including bars, nightclubs, and restaurants. Any public place where people wanted to listen to music. RFIDJs wanted to create a modernized version of the jukebox to bring back that nostalgic feeling for older audiences and at the same time, teaching the younger audiences what a jukebox was. Our design consists of ten RFID cards each assigned a specific song from spotify which is similar to the tracks of the original jukebox. Just like the original version, our design has a robotic arm that moves up and down, and smoothly grabs a RFID card with a robotic claw. We made our robotic arm to remember which card it took so that when the user wants to return the card, the RFID jukebox will automatically go back to the slot the card was returned. We also implemented a pushbutton control like the original jukebox to select which RFID card the user wants but we also implemented a voice control mode that once activated will wait for the user to say “hello” and “play track x”. Once the voice control recognizes the number the user asked (1 through 5), then the robotic arm will automatically move towards it and retrieve it. Once the card is grabbed in either mode, it’s taken down to an RFID scanner where the card’s unique ID is read and sent to an Arduino Uno (the hardware controlling the motors and mode selection).

II . PROBLEM DESCRIPTION

Throughout the process of building the RFID jukebox, we encountered many challenges whether mechanical issues, hardware issues or software issues. For our mechanical issues, we wanted our jukebox to have a robotic claw that moves up and down with a stepper motor. This stepper motor would be attached to a railing system that could be mounted on the base of our project. At first, the stepper motor alone was able to move up and down the railing system smoothly, but as we added more and more weight to the moving platform of the railing system, the robotic claw, it would cause the platform to bend down due to gravity and the weight of the claw which evidently, caused friction between the platform and the metal rail. Because of the friction being generated, the robotic claw would jam while going up or down and made a screeching sound. For our hardware issues, we needed to take into consideration how many IO pins we needed to use for all of the jukeboxes components. And lastly, our software issues consisted of trying to install the proper libraries to get Spotify API to work on the rock Pi, to control the stepper motor’s movements smoothly and to have a working connection between the Rock Pi and Arduino Uno through its TX/RX pins.

III. SOLUTION

A. Proposed Solution

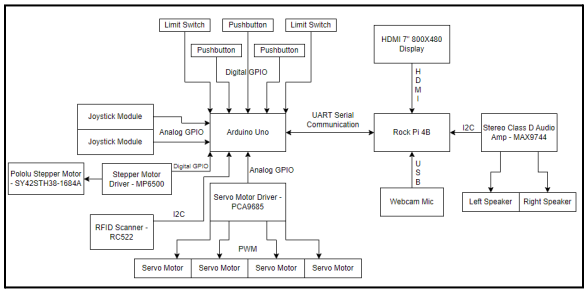


Figure 1: Hardware Implementation Block Diagram

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

Our project's hardware design features a Rock Pi 4B and an Arduino Uno. The Rock Pi 4B will serve as our main computer and will help in times when we need internet connectivity. The purpose of the Arduino Uno is to work as an external I/O board. Having the Arduino here is advantageous because it frees up resources on the Rock Pi. While the Rock Pi is faster for tasks that require more memory and processing power, the Arduino is also better for real-time processing and low power consumption. We also wanted to distribute our tasks so that we lower the risk of either board overheating.

In our project's hardware design, seen in figure 1, we had several major functions that we wanted accomplish, each involving different components:

1. Ability to move RFID cards
    - Sunfounder Servo motor arm
    - PCA9685 Servo motor driver
    - NEMA Stepper motor
    - PCA9685 Stepper motor driver
2. Ability to scan and uniquely identify cards
    - RC522 RFID Scanner
3. Mode and card selection with pushbuttons
    - Pushbuttons
4. Card selection with joysticks
    - Joystick modules
5. Card selection with voice recognition
    - Webcam Microphone
6. Ability to send packet from Arduino to Rock Pi
7. Ability to play music from our Pi
    - Class D Audio Amp
    - Speakers

Regarding our ability to move RFID cards to and from the scanner, we implemented this with several parts: sunfounder servo motor arm, the PCA9685 servo motor driver, a NEMA stepper motor, and a stepper motor driver. The sunfounder arm shown in figure 2 has 4 servo motors, each responsible for a different movement. One motor controls the gripper, another the wrist, an elbow motor, and one for the shoulder. These 4 servo motors are controlled by our PCA9685 servo motor driver, which allows us to direct the movement of the arm with a single I2C interface. This frees up space on our only need to use 2 analog pins. It also includes a built-in clock to ensure precise timing of the PWM signals, which helps to improve the accuracy of the servo control.

Figure 2: Sunfounder Servo Motor Arm

The vertical movement of our arm is controlled by a stepper motor rail. On this rail we have a base attached to a threaded pole, which is attached to a stepper motor. When the stepper motor spins this pole, the base will be guided up or down by the threads on this pole. This configuration allows us to convert the rotary motion of the motor into linear motion on the rail. Systems like this are used for CNC machines, 3d printers, and other robotic systems. The movement of our arm is not enough to cover the full distance of our card rack, so we chose to have a rail like this and the build allows for high accuracy, repeatability, and holding torque. The last of these isn't as important as our build won't be subject to large forces but accuracy and repeatability will be important for the repeated movements of our arm.

The ability to scan and uniquely identify cards is handled by our RC522 RFID scanner. This scanner uses radio waves at 13.56 MHz to read information from RFID tags or cards. The device can be programmed from an Arduino using the MFRC522 library. While usually used for applications such as access control, inventory tracking, and payment systems, the use of RFID cards in our project provides a fun way to display songs and then communicate to our computer which song was selected.

The mode and card selection with push buttons will be one of our simpler functions to implement and will be the first hardware that the user will interact with. We plan on having 3 stand-alone push buttons on digital GPIO pins, with another two coming from our joystick modules. The 3 solo push buttons will be

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

used in selecting one of the 3 modes of operation for our project and all 5 will be used to select from our 5 displayed cards.

We plan on using Adafruit's Analog 2-axis Thumb Joystick with Select Button + Breakout Board. Each joystick module has two potentiometers and a push button. The potentiometer acts as a voltage divider, and the analog input of the Arduino can measure the voltage at the wiper (output) of the potentiometer to determine the joystick position. With the information on the joystick position we can know what the user is signaling for. X and Y on the first joystick can control the claw and wrist on our arm, respectively. X and Y on the second joystick can control the elbow and shoulder. The two push buttons can control the vertical movement of the base on the stepper rail.

We plan on implementing the serial communication between the Arduino and the Rock Pi in a similar fashion to how our computer engineering CSE398 color sensor project was set up. One difference would be that we would use the MRAA library to read data from our Arduino, rather than using a C++ library. This makes things simple as we are already using python in our other Rock Pi functions. Serial communication from the Arduino is easy as there is already a built-in serial interface. A packet can be constructed using the PacketSerial library. Data can be sent just by writing to the right serial interface. We would also have to make sure that both devices have the same baud rate, parity bits, etc.

The Rock Pi already has a speaker that is able to play music but we chose to connect external devices to get louder, better quality sound for the sake of having a complete project. We chose the Adafruit 20W stereo class d amplifier because it ticked all of our boxes. It's got enough output power to drive our larger speakers when connected to another power source. It's got volume control. It has thermal protection and over-current protection built-in. It's also cheap, coming in at $19.95. Our Visaton BG 17 speakers were coincidentally ordered by Dr. Graham recently and fit our criteria.

We will be maximizing the use of the pins on our Arduino, the pinout shown in table 1. Pins 0 and 1 will be used to communicate with the Rock Pi with pin 1 being TX and pin 0 being RX. Pins 3 and 4 will be used for the stepper motor driver. Pins 2 and 10 to 13 will be used for the RC522 RFID scanner. Pins 5 and 6 will be used for limit switches. Finally, digital pins 7, 8, and 9 will be used for pushbuttons.

On the analog side, pins A0 to A3 will be used for our two joysticks and pins A4 and A5 will be used for our servo motor driver.

Table 1: Pinout on the Arduino Uno for the Whole Project

Pin Number	Assignment
A0 - A3	Joysticks
A5 - A4	Servo Motor Driver
0 -1	Serial Communication (RX & TX)
2	RFID scanner
3 - 4	Stepper Motor Driver
5 - 6	Limit Switches
7 - 9	Pushbuttons
10 - 13	RFID Scanner

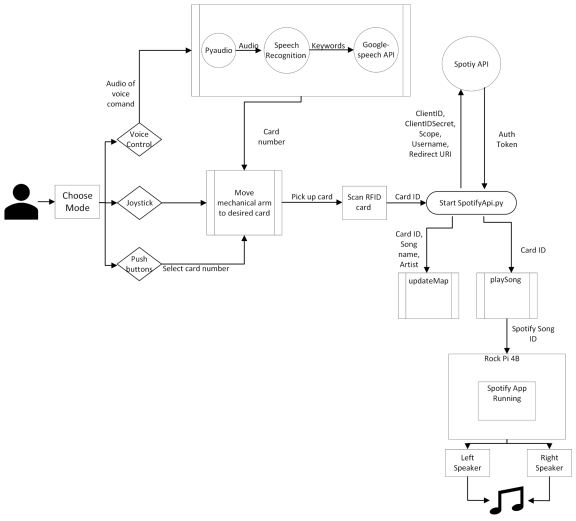


Figure 3: Software Implementation Flowchart

In order to incorporate musical functionality with the ability to search and play tracks stored on the RFID tags, we made use of the Spotify API. Included above in figure 3 is a flowchart of our implementation of the software we will be using. Our spotify code is written in python and we found that

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

there is a python library, spotipy, available that allows us access to all of the music data provided by the Spotify platform. It was created for use of the Spotify API with python. The spotify API is user based meaning that prior to using and gaining access to the API one must create a spotify premium account, log in and accept the latest Developer Terms of Service. This completes the account set up. After creating a premium account, we created an app on our developers site. This app will give you access to a client ID and client ID secret needed to connect to the API. Your app dashboard will allow you to keep track of the number and types of API calls you have made with the app and the client ID. You can see the app in figure 4 and the dashboard in figure 5.

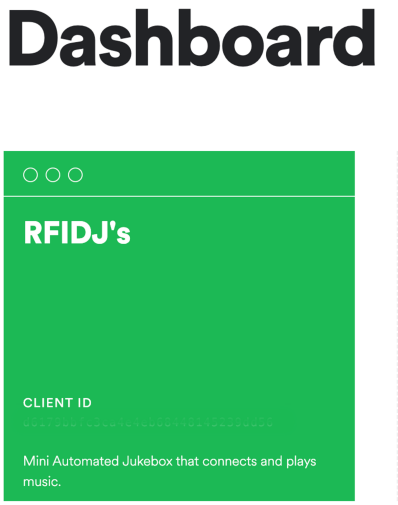


Figure 4: RFIDJ's App

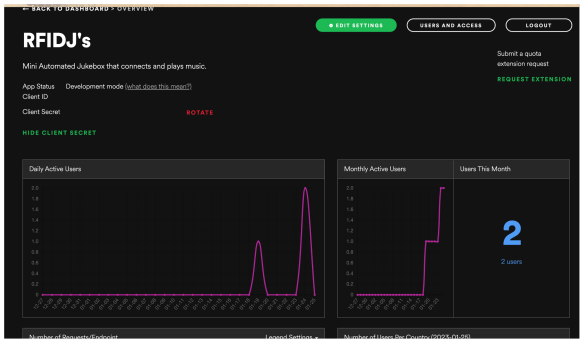


Figure 5: App Dashboard

After creating an app, we gained access to the API. We imported the following packages with Python into our python script, shown below in figure 6.

```
import requests
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from spotipy.oauth2 import SpotifyOAuth
import spotipy.util as util
from decouple import config
```

Figure 6: Python Imports

Note - For our project, in order to gain access to the requests and spotify packages we had to do a "pip install requests/spotify/decouple" separately to install the packages onto the device before importing them.
The following keys are necessary for proper connection to the spotify API:

- Client ID - This is obtained from the users app dashboard and is the public identifier of the app.
- Client ID Secret - This is obtained from the users app dashboard, it should be kept confidential and it is used to authenticate a users app.
- Username - This is the username of your spotify premium account, spotify needs this to retrieve specific information about songs, playlists and playback information
- Scope - This is explained here
- Redirect URI - This is the location where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token.

Note - To properly keep our client id, client id secret, username, scope and redirect uri hidden and secure (as those are keys that should not be accessed by just anyone), we followed the steps in this manual to create a hidden file to store those variables for later reference.

After successfully importing the right packages and securing our keys, we started working on our script. For this project, we created a SpotifyAPI class, with our initialization token derived in our constructor and a default empty song map with the RFID id as the key and spotify song id as the value (all the values are set to empty strings initially as the songs will need to be programmed onto the RFID card at first). The code for the SpotifyAPI class is seen below in figure 7.

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

```
class SpotifyApi:

    def __init__(self):
        CID = config('CID', default='')
        CID_SECRET = config('CID_SECRET', default='')
        SCOPE = config('SCOPE', default='')
        USERNAME = config('USERNAME', default='')
        REDIRECT_URI = config('REDIRECT_URI', default='')

        token = util.prompt_for_user_token(USERNAME, SCOPE, CID, CID_SECRET, REDIRECT_URI)
        self.sp = spotipy.Spotify(auth=token)

        self.songMapping = {
            1: "",
            2: "",
            3: "",
            4: "",
            5: "",
            6: "",
            7: "",
            8: "",
            9: "",
            10: "",
        }
```

Figure 7: Spotify API Constructor

Our class contains two functions; updateMap, that updates an RFID tag with the new song to be programmed on it and playSong, that obtains the song stored on an rfid tag and plays it on spotify. The code for updateMap can be seen in figure 8, the code for playSong can be seen in figure 9, and the main function can be seen in figure 10.

```
    """
    This function updates an RFID tag with the new song to be programmed on it
    @param rfidID: This is the ID of the rfid tag to be updated
    @param songName: This is the song name to be programmed onto the card and used in spotify search
    @param artist: This is the artist of the desired song and used in spotify search

    @return None
    """
    def updateMap(self, rfidID=None, songName=None, artist=None):
        for i in range(0,100,50):
            track_results = self.sp.search(q='track:'+songName, type='track', limit=50,offset=1)
            for i, t in enumerate(track_results['tracks']['items']):
                if t['artists'][0]['name'].lower() == artist.lower() and t['name'].lower() == songName.lower():
                    self.songMapping[rfidID] = t['uri']
```

Figure 8: updateMap Function

```
    """
    This function obtains the song stored on an rfid tag and plays it on spotify
    @param rfidID: This is the ID of the rfid tag with the desired song to play

    @return None
    """
    def playSong(self, rfidID):
        try:
            track = self.songMapping[rfidID]
            devices = self.sp.devices()
            print(devices['devices'][0]['id'])
            playTrack = self.sp.start_playback(device_id=devices['devices'][0]['id'], uris=[track])

            #return playTrack
        except:
            print("An exception occurred")
```

Figure 9: playSong Function

Note - in order to get active devices, we made sure that spotify was running as either a desktop application or a web application and that you were signed in as the user you specified.

```
# MAIN
if __name__ == "__main__":
    sp = SpotifyApi()

    sp.updateMap(1, "Umbrella", "Rihanna")
    sp.updateMap(2, "Wildest Dreams", "Taylor Swift")
    sp.updateMap(3, "Work Out", "J. Cole")
    sp.updateMap(4, "Gone Girl", "SZA")
    sp.updateMap(5, "Traitor", "Olivia Rodrigo")

    b = sp.playSong(3)
    print(b)
```

Figure 10: main Function

For the voice recognition portion of our project we plan to utilize the Pyaudio python package. It provides Python bindings for PortAudio v19, the cross-platform audio I/O library. In addition to that, we will be using the Speech Recognition package and the google-Speech API client. All of these are packages available in python that must be installed through pip before use. The google-speech API will give us real time voice recognition capabilities, with the option to visualize the output. During the voice recognition section of our project, we will look for keywords that will tell us what to do. Keywords such as “play”, “track”, or any sort of number would work. For example, a command such as, “Play track 5”, will allow the mechanical arm to move to the 5th RFID card, place it on the scanner and allow for the spotify song programmed onto the card to play. We made use of this reference to understand how to use speech recognition and the google-speech API for our project.

B. Designing the Modernized Jukebox

In the design of the outer parts of our jukebox we had two things to consider mainly: how our hardware would fit together and what materials were available to us.

One of our first design considerations was how the cards would hold and how our robotic arm would be fashioned around them. We used the same two color high density polyethylene (HDPE) and CNC milling machine that we have in the lab to design and cut out the pieces that we needed to construct our card rack that holds 5 RFID cards. The base of our rack has a similar design to the HDPE base that holds our

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

stepper motor and it is engraved with our team name (RFIDJs).

We situated both the arm and the card rack on one of our HDPE workstations that already existed in the lab when testing the movement of our arm for our fall semester hardware demonstration. We decided to stick with this setup as it was perfect for our arm's range of motion, compact, and used materials that we already had in the lab.

The HDPE in the lab continued to be useful in the design of our jukebox, alongside the polycarbonate that was also available to us in the EECS Shop. With the help of Tim O'Breen we made the outer casing of these two materials. This case, which consisted of an upper and lower part, housed the two speakers and 7" LCD screen in the upper part (made of HDPE) and the rest of the project in the lower part (made of polycarbonate). A sketch of the lower half design is shown in figure 11. The lower half being made of this transparent polycarbonate allows the user to see all of the moving parts of our project clearly.

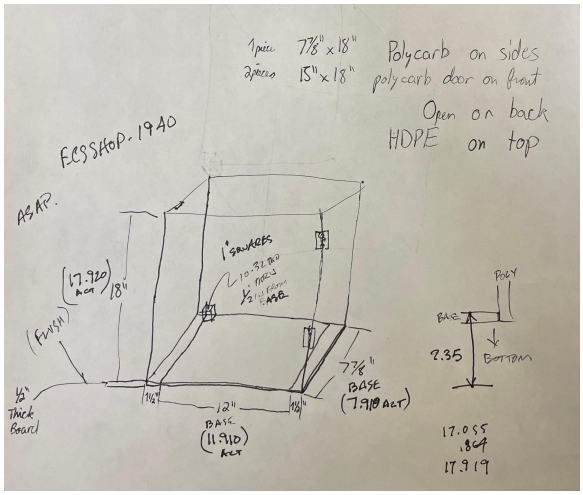


Figure 11: Sketch and dimensions for the lower part of our frame

To put some finishing touches on our design we 3D printed a video game console style controller to hold our joysticks and a box that could be attached to the HPDE workstation to hold our pushbuttons.

C. *Setting Up the Stepper Motor*

The overall robotic arm system consists of the Pololu MP6500 stepper motor driver, a pre-built railing system seen in Figure 12, an Arduino Uno, two limit switches, three pushbuttons, and a perfboard to solder the system and mount it inside the robotic claw. All of these materials were found in the lab room, the only thing that our team ordered was the Claw kit.

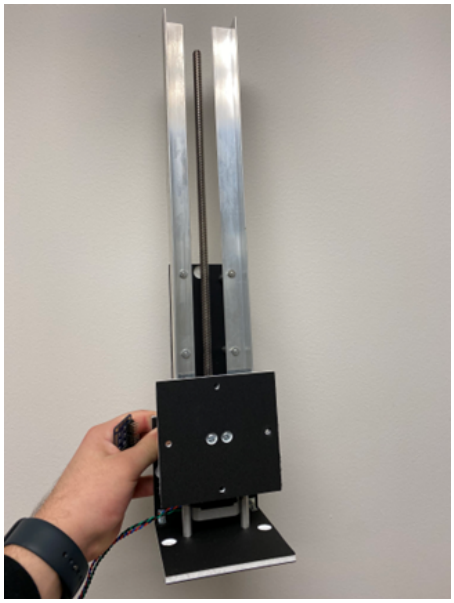


Figure 12: Prebuilt railing system found in the lab where we mounted the robot claw and what we'll use as the mechanical arm inside the jukebox.

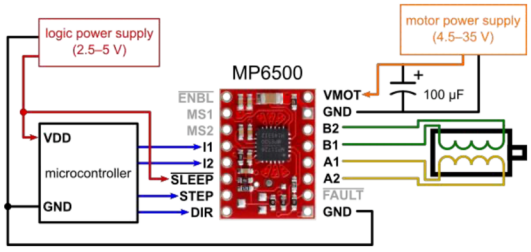


Figure 13: Wiring diagram of the stepper motor driver

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

Using the diagram in figure 13, the connection between the stepper motor, arduino uno and the driver was organized and documented on the following table in figure 14 so that the process of soldering each wire to a perfboard would be an easier process.

Motor Driver pin	Assigned Perfboard pin	Arduino Uno pin	Assigned Perfboard pin	Other pins	Assigned pin
VMOT	D1	Digital PWM 9	H8	Perfboard Left +	H6
GND	D2	Digital PWM 8	H7	Perfboard Left -	C8
B2	D3	Power 5V	Left +	Up button blue wire	Arduino Uno Digital PWM 2
B1	D4	Power GND	Left -	Up button white wire	Perboard Left -
A1	D5	**Stepper Motor Wire**	**Assigned Perfboard pin**	Down button red wire	Arduino Digital PWM 3
A2	D6	Blue	C6	Down button white wire	Perfboard Left -
FLT	D7	Red	C5	Limit Switch red wire	Arduino Digital PWM 4
GND	D8	Green	C4	Limit Switch white wire	Perfboard Left -
EN	G1	Black	C3	**Servo Motor Driver**	**Assigned Perfboard pin**
MS1	G2	**Capacitor Pins**	**Assigned Perfboard pin**	GND	J24
MS2	G3	+ pin (longest one)	C1	OE	J23
I1	G4	-pin (shortest one)	C2	SCL	J22
I2	G5	**5V power Supply**	**Assigned Perboard pin**	SDA	J21
SLP	G6	+ (red)	Right +	VCC	J20
STEP	G7	-(black)	Right -	V+	J19
DIR	G8				

Figure 14: Wiring table of stepper motor, arduino uno, and motor driver to perfboard.

After wiring and soldering the appropriate wires to its designated port and position at the perfboard, we utilized the AccelStepper.h library on our Arduino code in order to accomplish smooth movements up and down for the robotic claw. With the library, we made the motor rotate 1300 steps clockwise or counterclockwise (up or down). Another reason we used the AccelStepper.h library was for the ability to use the stop( ) command. This proved to be very useful to our design to implement safety features like the two limit switches that prevent the robotic claw to an excessive amount up or down.

To improve the stepper motor movement, Ibrahima and Gilberto took the jukebox to Link Hall and had Professor Tim Breen look at it. The issue we had was that the weight of the claw was making the platform hang down and rub against the metal rail structure of the railing system. Meaning that as the arm moved up and down creating friction, it made the robotic claw jam up, causing a loud noise and the robotic claw wouldn't reach the next platform before stopping. To counter this, Professor Breen machined these white pieces, in figure 15, made of a non-stick material (meaning that it will slip on metal) and installed it right on the platform so that it is pinned between the metal rail and the platform holding the robotic claw.

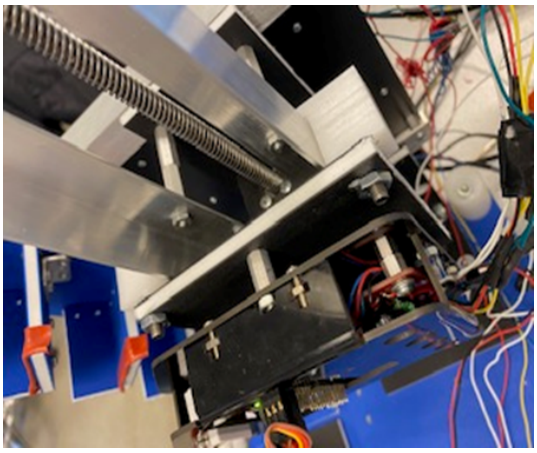


Figure 15: White braces added to improve stepper motor movement

D. Setting Up the Servo Motors on the Mechanical Arm

For the mechanical arm of the jukebox, we made the decision of purchasing a robotic arm kit, the Sunfounder Servo motor arm to be more specific. We made this decision to make the process faster of building a working robotic arm by usings its servo motors and overall structure. But the robot kit has four servo motors in total and we can't use them individually on the Arduino uno because there aren't enough I/O pins. To solve this problem, we wired all the servo motors together on a PCA9685 Stepper motor driver with the following wiring:

Arduino 5V -> Servo Motor Driver Vcc
Arduino GND -> Servo Motor Driver GND
Arduino Analog Pin 4 -> Servo Motor Driver SDA
Arduino Analog Pin 5 -> Servo Motor Driver SCL
Servo Motor 1 (Shoulder Twist) -> Servo Motor Driver Pin 0
Servo Motor 2 (Shoulder Up/Down) -> Servo Motor Driver Pin 4
Servo Motor 3 (Elbow) -> Servo Motor Driver Pin 8

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

Servo Motor 4 (Claw) -> Servo Motor Driver Pin 12

The following figure, figure 16, will better demonstrate the wiring between the servo motor driver and the arduino uno.

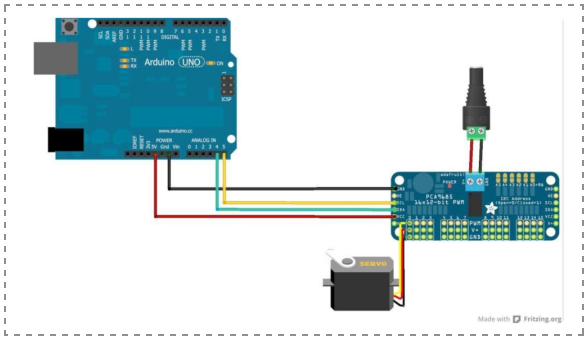


Figure 16: Servo Motor driver wiring diagram

Once the robotic arm was built and all the motors were wired, we implemented the system to the same perfboard that the stepper motor is wired at to combine the two systems and have it all powered by a single source. The final look of the mechanical arm can be seen in figure 17.

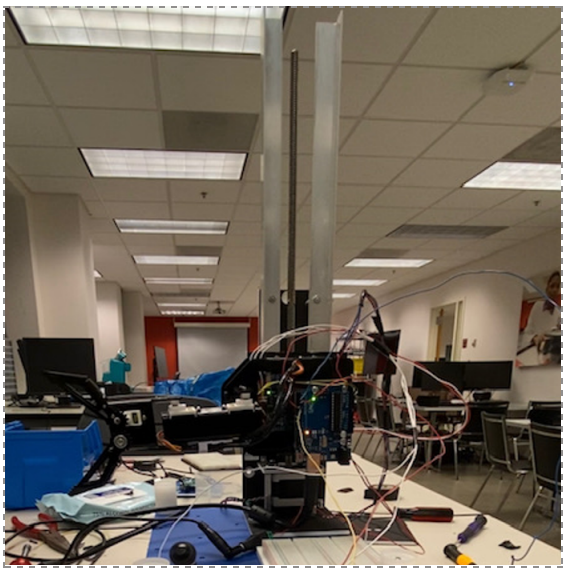


Figure 17: Completed Mechanical arm

An issue we would deal with once the first version of the code to control the servos was complete was that it would move extremely fast from one angle to the other which in turn caused the servo motors to freak out and jerk up and down in a crazy manner. To fix this physically, we changed the voltage source from the regular brick to a powerful voltage source box that can provide a constant 5.5 volts at a very high current of amps. But the biggest difference was in the software. Before, we used the Adafruit_PWMServoDriver.h library and just sent a simple PWM signal to the servo motor. It's because of that implementation that the servo motor moved so fast, but now, we upgraded that method by using the same library but created a new function from scratch. A function that uses the current angle and increments it one by one to the angle goal with a delay on a for loop. The implementation of the code is seen in figure 18.

```
void moveServo(int delayTime, int currentAngle, int angleGoal, int servonum){

  if(currentAngle < angleGoal){
//      Serial.println("++");
//      Serial.println(currentAngle);
    for (uint16_t pulselen = pulseWidth(currentAngle); pulselen < pulseWidth(angleGoal); pulselen++){
      delay(delayTime);

      pwm.setPWM(servonum, 0, pulselen);
    }
  }
  if(currentAngle > angleGoal){
//      Serial.println("-- NEGATIVE");
//      Serial.println(currentAngle);
    for (uint16_t pulselen = pulseWidth(currentAngle); pulselen > pulseWidth(angleGoal); pulselen--){
      delay(delayTime);

      pwm.setPWM(servonum, 0, pulselen);
    }
  }
  delay(100);
}
```

Figure 18: Code implementation to control servo movement

With this new method, the servo motors can now be controlled on how slowly they can be moved by how big the delay variable called “delayTime” is. With the upgraded servo movement, the jukebox was now more reliable to move and grab each RFID card.

E. Getting Push Buttons to Control the Movement of the Stepper and Servo Motors

For one of the modes in the jukebox to choose and play an RFID card, we implemented a push button

mode. We soldered three push buttons (two black buttons and one red button). The specific buttons used are known as Chassis buttons and were acquired from the lab room. Each button consists of two prongs and to wire it to the system we made the following connections:

Wire 1 on button -> ground
Wire 2 on button -> GPIO pin

To implement the buttons in code, we implemented the ezButton.h library on arduino and made a function that is constantly checking if a button is pressed or not. The format of the code is seen on figure 19.

```
int checkPush(const char pinNumber, bool buttonState) { //This function determines which button is press

  bool buttonPushed = digitalRead(pinNumber);


  if ((buttonPushed == buttonState) && (pinNumber == grabButtonPin)){
    Serial.println("grab button");
    userMode = grabCard(armLevel);
    armLevel = armCounter(pinNumber, armLevel, true);
    delay(450);
    while(digitalRead(pinNumber) == buttonState){
    }

  }

  if ((buttonPushed == buttonState) && (pinNumber == upButtonPin)){

    armLevel = armCounter(pinNumber, armLevel, false);
    Serial.print("Going up to level ");
    Serial.println(armLevel);
    moveMotor(distance);
    delay(450);
    while(digitalRead(pinNumber) == buttonState){
    }
  }


  if ((buttonPushed == buttonState) && (pinNumber == downButtonPin)){

    armLevel = armCounter(pinNumber, armLevel, false);
    Serial.print("Going down to level ");
    Serial.println(armLevel);
    moveMotor(-distance);
    delay(450);
    while(digitalRead(pinNumber) == buttonState){
    }
  }
  return userMode;
}
```

Figure 19: Pushbutton code

What this function does is to check all three buttons at the same time to see if one is pressed. If either of the black buttons were pressed (depending on which one) it would call the function that moves the stepper motors up or down. We formatted the code to stop checking for a push button press while the robotic arm was moving, so the user had to press the button, wait for the motor to reach its destination and press it again. For the red button, once this is pressed, it calls the function that makes the robotic claw automatically reach and grab the RFID card and takes it all the way down to the scanner.

F. Setting up the RFID Card Scanner With The Arduino Uno

The RC522 can communicate with I2C, SPI, or UART. We will be using SPI because it's the fastest, has no addressing system, and can transmit continuously without interruption.

To prepare the scanner for use in this project we needed to identify where it would fit in and properly connect it. We are using an arduino uno as an I/O board so connecting the RC522 scanner to the arduino made the most sense. It's also useful to connect the scanner here because the scanner does take up a lot of pins.

Below is a quick rundown of each pin on the scanner and where we have it connected on the Arduino:

-VCC is the pin that will power the module and we have it connected to the 3.3V pin on the arduino

-RST is used for powering down and resetting the whole module, we have it connected to Pin 9.

-GND is the ground pin and is connected to the Arduino's GND pin.

-MISO/SCL/Tx will be our MISO pin in our SPI communication, we have it connected to Pin 12.

-MOSI is the SPI input and we have it connected to Pin 11.

-SCK is the serial clock pin and will be connected to Pin 13.

-SS/SDA/Rx is the signal input pin for our SPI comm. and is connected to Pin 10.

Each RFID card contains an 8 byte unique ID (UID) that we can send to the Rock Pi 4b. This representation can be seen in figure 20.

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

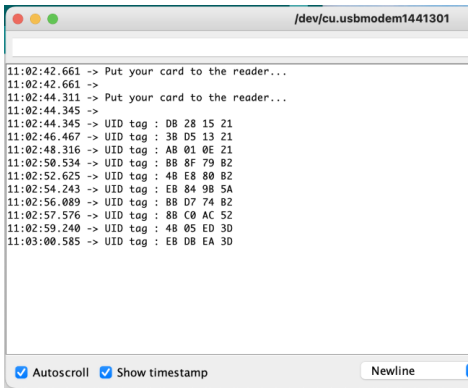


Figure 20: The output of our RC522 Scanning Text.

G. Getting the Spotify API working on the Rock Pi

For this process, the first thing we did was to develop the code with the working spotify API connection on a separate computer. This was so that we could ensure that we knew a way to make a successful connection to the API and test the different songs that we had in mind before further transferring it to the Rock Pi. It was through this that we discovered the spotipy library in python that acts as a Python Library for the Spotify API, so instead of connecting to it directly, we did it with Spotipy. Whilst developing this code, we were able to retrieve the specific keys that we required and store them in a secret file. In order to play a specific song, the API requires the song title and artist name. After feeding the information of the songs we wanted to program onto our RFID cards, the next step was to filter a lot of the data that we received to make sure that we were getting the right song ID to use when instructing the API for which song to play.

H. Writing a Program to Map the RFID Card ID With the Song ID

After obtaining the song ID from the Spotify API, the next step was to map it to an RFID card. This was an essential part of our project as it represented the way in which the tracks are stored and how users could differentiate them. To make each Card ID to a song ID, we created two separate python dictionaries; one with the Card Number as the keys and Card RFID ID as the values and the other with the Card Number as the key and Song ID as the value.

```
self.idMap = {
    1: "DB 28 15 21",
    2: "3B D5 13 21",
    3: "AB 01 0E 21",
    4: "BB 8F 79 B2",
    5: "4B E8 80 B2",
    6: "EB 84 9B 5A",
    7: "BB D7 74 B2",
    8: "1E FC C9 10",
    9: "4B 05 ED 3D",
    0: "EB DB EA 3D"
}

self.songMapping = {
    1: "",
    2: "",
    3: "",
    4: "",
    5: "",
    6: "",
    7: "",
    8: "",
    9: "",
    0: ""
}
```

Figure 21: This was the initialization of both dictionaries

The songMapping dictionary gets updated with a song ID in our update function that searches for a specific song based on the artist name and song title and returns the spotify song ID for that song (the Spotify Song ID is what Spotify uses to uniquely identify each track).

```
sp.updateMap(1, "Rock that Body", "Black Eyed Peas")
sp.updateMap(2, "Boy's a liar", "PinkPantheress")
sp.updateMap(3, "Waiting For Love", "Avicii")
sp.updateMap(4, "Vete", "Bad Bunny")
sp.updateMap(5, "Traitor", "Olivia Rodrigo")#
```

Figure 22: This is a snippet of how our updateMap(Figure 8) function is used

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

When a card is picked up by the mechanical arm and has its RFID ID (UID) read, we check what card number that ID has been mapped to in our idMap dictionary (either 0-9). After that card number is returned from our getID function, we use it to retrieve the spotify song ID that is mapped to that card.

```
def getID(self, UID):
    n = ""
    for num, uid in self.idMap.items():
        if uid.replace(" ", "") == UID.replace(" ", ""):
            n = num
    #print(n)
    return n
```

Figure 23: getID function that returns what Card number is mapped to a UID

The spotify Song ID is then passed to the play song function that plays that song on an available device shown on the spotify users account.

```
def playSong(self, rfidID):
    #try:
    track = self.songMapping[rfidID]
    devices = self.sp.devices()
    print(devices['devices'][0]['id'])
    playTrack = self.sp.start_playback(device_id=devices['devices'][0]['id'], uris=[track])
```

Figure 24: playSong function

I. Setting Up the Speakers With the Rock Pi

At the beginning of the process, we researched a lot on which speakers to use, how to wire them, and how we can implement it to the jukebox so that it'll be used. But overall, this is what we needed to accomplish in terms of wiring:

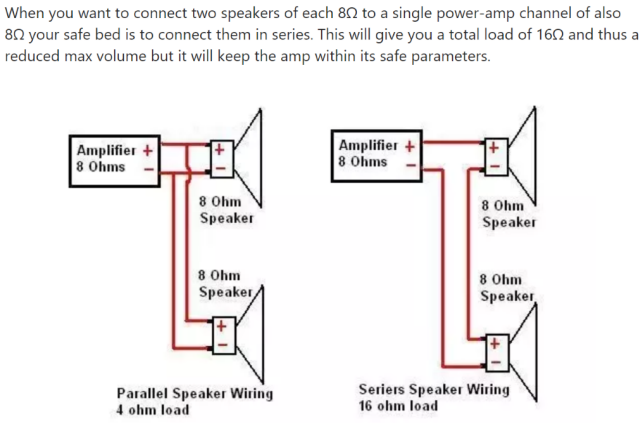


Figure 25: Summarized wiring of the speakers

Luckily, the lab room had just ordered new speakers for students to use and we got 2 BG-17 Visaton speakers. In addition to the speakers, we needed an amplifier that could power the two speakers at the same time. Which is why we went with the Stereo 20W Class D Audio Amplifier, MAX9744 Adafruit with the following capabilities:

- Drive 2 channels of 4-8 ohm impedance speakers at 20W each
- Power with 5-12V DC
- Analog & digital volume control capability
- 20mA current

To prepare the audio amplifier, we needed to follow the assembly tutorial and soldered the different terminals and capacitors onto the audio amplifier board.

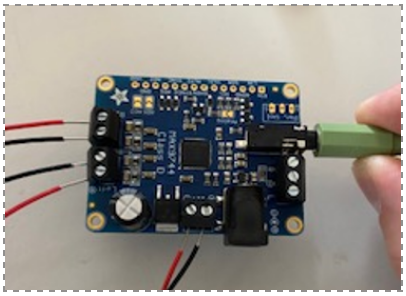
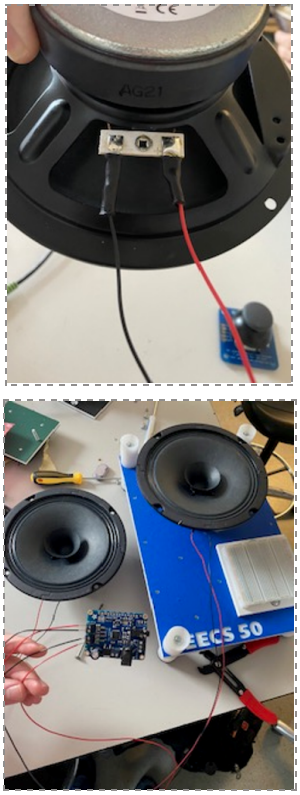


Figure 26: Completed audio amplifier

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

Once the amplifier was completed, all we had to do was solder some wires onto the speakers and hook it up to the speaker terminals of the amplifier.



Figures 27 & 28: Completed speaker system

A benefit to the Class D amplifier is that it comes with an audio jack, meaning that all we needed to do to use it was to connect it to the rock pi or any device that will be playing the song with an audio cable.

J. Writing a Program For the Voice Control

To complete the voice control aspect of this project we required a python library that allows us to connect to either internal or external microphone(s). This is because we needed to stream the input from the microphone and analyze/parse it for specific keywords and/or phrases. Similarly to how we started developing the spotify API code, we developed this on a separate computer to test it before later transferring it onto the Rock Pi. To achieve the goal of this Voice Recognition section, we utilized the SpeechRecognition library in python. It is a library that supports several speech recognition engines such as Google Speech Recognition, Sphinx, and Wit.ai. Using SR, we streamed the voice from the microphone and used the Google Speech recognition engine/API to recognize what was being said. We decided to use this engine in comparison to the others available under the library due to a few factors:

- Familiarity with the engine, a group member had made use of this before.
- Popularity and Accuracy, as it is the engine most people tend to use more in comparison to the others and it is one of the most accurate speech recognition engines available.
- Cost effective, in comparison to some of the others, It is free to use with no set limits.

The great thing about the Google speech engine is that, after input from the microphone is passed into it, it returns a set of possible matches for what the user has said, along with the confidence it has for that specific recognition.

```
r = sr.Recognizer()
```

Figure 29: Initializing the Speech recognition object

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

```
with sr.Microphone() as source:
    print('Please start speaking...\n'
    while True and startUpAttempts !=
```

Figure 30: While loop that listens in for Microphone input

```
try:
    print("hey")
    text1 = r.recognize_google(audio1
```

Figure 31: Using the Google Speech Engine in our code

To ensure that our program was not just running continuously and picking up different words and dialogue, we established a sort of “wake” word in our code. After the user chooses the voice control mode, the program starts listening for the keyword, “hello” (our “wake” word). The program listens for about 5 seconds and if no command is provided or “hello” is not heard, then the program loops back to listen again. This will work 3 times, before the program exits as the user gets three tries/attempts. This is to allow for us to catch some unexpected errors that may occur while the program is listening in. We used this same process to retrieve the “play track command” after the user has mentioned the wake word. Once the program detects the phrase “play track”, it exits and returns the number detected after the phrase as the card number, so the mechanical arm knows which card to grab.

```
with sr.Microphone() as source:
    print('Please start speaking...\n')
    while True and startUpAttempts != 3:
        print("heey")
        startUpAttempts += 1
        audio1 = r.listen(source, 6)
        try:
            print("hey")
            text1 = r.recognize_google(audio1)
            if keyWord1.lower() in text1.lower():
                playCommand = True
                print('Start up Keyword detected in the speech.')
                print('Please give play command.')

                while playCommand and attempts != 3:
                    attempts += 1
                    audio2 = r.listen(source, 5)
                    try:
                        text2 = r.recognize_google(audio2)
                        if keyWord2.lower() in text2.lower(): #and keyWord3.lower() in text2.lower():
                            print('Keyword detected in the speech.')
                            print(text2)
                            playCommand = False
                            startUpAttempts = 3
```

Figure 32: Voice Control Sequence

```
                text2_list = text2.split()
                trackSelect = text2_list[-1]
                print(trackSelect)
                if trackSelect.lower() == "one":
                    trackSelect = 1
                elif trackSelect.lower() == "two":
                    trackSelect = 2
                elif trackSelect.lower() == "three":
                    trackSelect = 3
                elif trackSelect.lower() == "four":
                    trackSelect = 4
                elif trackSelect.lower() == "five":
                    trackSelect = 5
                print(trackSelect)

                try:

                    trackNum = int(trackSelect)

                    print(trackNum)
                    return trackNum
                except ValueError:
                    print("No valid integer following /'Play track/'")
                    return -1
```

Figure 33: Voice Control Sequence

```
Please start speaking..

result2:
{   'alternative': [{'confidence': 0.98762906, 'transcript': 'hello'}],
    'final': True}
Start up Keyword detected in the speech.
Please give play command.
result2:
{   'alternative': [   {'confidence': 0.8327499, 'transcript': 'play track 5'},
                       {'transcript': 'play track five'}],
    'final': True}
Keyword detected in the speech.
play track 5
```

Figure 34: Voice Recognition transcript with recognition of “wake” word and “play track” phrase

As shown in figure 34, the results provided consist of an alternative section, which entails possible alternative options for what the user has said. This consists of a transcript, the actual diction and a confidence. In addition to alternatives there is a final section that concludes whether it has been able to recognize what has been said.

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

K. Sending RFID info from Arduino to Pi with UART protocol.

In connecting the two halves of our project we needed to choose a communication protocol to use between the Arduinon and Rock Pi. We used UART because it's simple to operate and well documented. The disadvantages of UART, such as inability to use multiple primary/secondary systems and low data transmission speeds, were also not relevant to us for this connection.

In our project proposal we had originally described that we would be sending COBS encoded packets from between the two boards but that is no longer necessary as the UID of each RFID card is not sensitive information and we can use logic on our Rock Pi to collect the bytes after our header without formally constructing a packet.

The hardware setup for this task is very simple. We needed to connect the TX and RX pins on each side. It is important to make sure that the boards share a common ground.

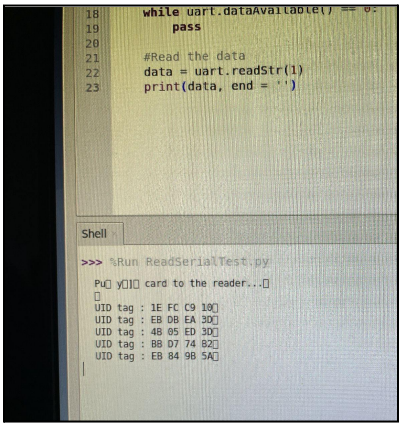


Figure 35: Reading UID tags from the Arduino on the Rock Pi.

IV. DEMONSTRATION

For our final demonstration RFIDJ's demonstrated how the modernized jukebox can be controlled two different ways. Our project was very engaging for the audience because it allowed the user to participate by testing the jukebox one of the two ways. One method was to use push buttons; the user was able to push the up and down buttons until it reached one of the five songs they chose, at which point they were able to hit the third button, which grabbed the RFID card and took it to the scanner to read and play the specific song they wanted. The second option was voice commands, which allowed the user to communicate with the jukebox to request a song. For example, if the user said "play track 3," the jukebox would automatically select and scan track 3. After the user controlled the mechanical arm in one of the two ways, and selected one of the five RFID cards on the shelf, the mechanical arm would place the RFID card on the scanner, and the song would be played from the sound system and displayed on a screen, which was Elizabeth's computer.

V. RESULTS

We wanted to make sure that control of our jukebox by voice commands would work properly not only in the lab, but also in a potentially crowded, loud open house setting. With this in mind, it was important tracked the confidence of our voice recognition system with different levels of background noise

Below are bar charts showing the recorded confidence of our voice recognition library on all of our command keywords. There are two sets, showing the difference between how confident our recognition is in silence vs with crowd noise at around 80 dB.

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

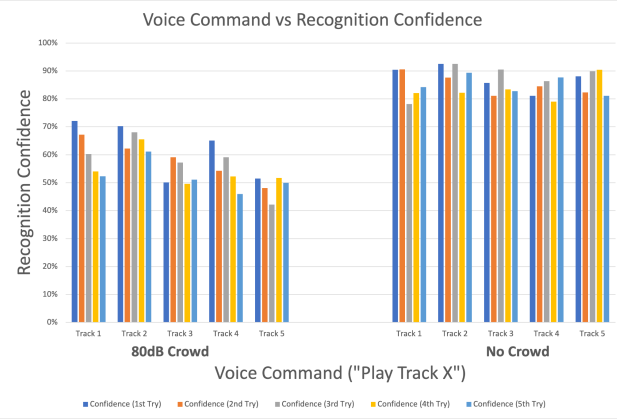


Figure 36: The confidence of our voice recognition system on different voice commands in two different settings.

The speech recognition library returns a number between 0 and 1 inclusive that represents its confidence on the phrase. As shown in Figure __, we were able to get an average confidence above 0.8 on every phrase without a crowd and above on most phrases 0.5 when simulating a crowd. Even on our least confident phrase we were able to manage an average confidence level above 0.45.

VI. CONCLUSION AND DISCUSSION

In this project we were able to create a working jukebox using RFID cards which were controlled by push buttons and voice recognition commands. To build this project, the use of RFID cards, RFID scanner, Arduino Uno, Rock Pi 4B, and many other hardware components. Our project enabled the user to easily choose a song either by using push button control and or voice commands. The RFID cards were to hold the specific songs and the scanner was to send the specific song ID to the Rock Pi 4B, while the arduino uno was used to control the motors. Both the Arduino Uno and Rock Pi communicated together. The voice commands allowed a hands-free control of the jukebox, which made it even easier to control than the push buttons. Overall, RFIDJs were able to design and build a jukebox from scratch using modern technology.

Although we were able to complete our project in time for the open house and the final demonstration, there were issues we ran into along the way.

The first major issue we ran into was figuring out the joystick control. Originally we proposed that we would have three ways of controlling the jukebox; push button control, voice commands, and joysticks. When implementing the joysticks into our project, we could not figure out how to code it in the Arduino IDE. We were able to implement them when the servo motors were connected to the Arduino Uno but we could not figure out the code for when the servo motors were connected to the servo motor driver. We tried coding in the joysticks a couple different ways in Arduino by changing the libraries, using the mapping function, and just reading and writing to them. Due to the open house and the final demonstration coming up, we decided to take this subtask out of our project and focus more on a working jukebox.

The second issue was fixing the servo motors after we tried implementing the joysticks. Once we tried getting the joysticks to work, the servo motors didn't work correctly with the push button control. To check this problem, we tried using the analog discovery to see if there was voltage going to the motor that was causing an issue. When we checked the analog discovery, there was voltage going to it but it was pulsing. A solution for this was trying to use a different power bank, one that had more power but had the same voltage. When we tested this, it didn't change much, the servo motor would work off and on. The reason the servo motor kept pulsing back and forth was because of something in our code. Since it worked before the joysticks, we realized we must've changed something in the code. After looking through the code and adding the new voltage source, the push button control for the servo motors worked again.

A third issue was with the voice recognition program. When running the voice command code, the program would understand the original command "hello" and play a specific track and move the mechanical arm to a card to grab it and scan it. But

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

after that it would not play the song on the Rock Pi. It would print out the song ID but not play it because the line of codes that asked the user to talk into the microphone to “wake up” the program and play a track was in a continuous loop, so it would never get to the code after it telling it to play the specific track. To solve this problem, we had to alter the code. In the loop we told it to look for a specific symbol and if it found it, it would loop continuously, but since the symbol it was looking for was in the print statements in our Arduino code, it would continue to loop. So in the end we had to change the symbol we used to one that wasn’t printed in any code.

Those were the only major issues we had with the project, all other issues were minor and we were able to fix them within a few minutes.

VI. REFERENCES

[1] https://www.tiktok.com/t/ZTRsEq9EG/

VII. APPENDICES

A. Appendix 1

1). Shall be primarily a computer and electrical engineering project. All projects will contain electrical and computer engineering components. You will have to show that your design is electrical and computer engineering focused, even when mechanical, aero, bio, chemical, civil, environmental, and/or computer science components are used: To satisfy requirement one, the modernized jukebox is essentially a project of computer and electrical engineering components. The project's electrical engineering components were to move the mechanical arm using push buttons. This entailed getting the stepper and servo motors to move in tandem with the push buttons. In addition, the sound system with two speakers and an amplifier had to be installed for the music to play out of the jukebox. With the Rock Pi 4B and Spotify API, the computer engineering components were configuring the RFID cards and scanner to work with the Spotify API. It also included writing the code for the voice recognition program. The only non-computer or electrical task was designing and building the jukebox shell and RFID card holder.

2).Shall include a reference to the most similar project found on the internet (YouTube, Instructables, TikTok, other University Senior Project pages, etc.) and a justification as to how your project is improved or different: For requirement two, we got the concept from a similar effort we saw on TikTok. The original concept was known as a Spotify Jukebox. The link will be in the reference section of this report. The developer attached a Raspberry Pi to an RFID scanner, which scanned RFID cards that he coded to play different songs. For our project, we will be extending his project with multiple user interface controls and including the option to choose any song meaning we'll program a new song onto RFID cards. Furthermore, our project will be substantially different from his design. We planned to make it look like a classic jukebox, but with modernized mechanical, electrical, and computer components. In the end due to the components not coming in on time, we just made the outer shell a box with a clear acrylic but the top where the speakers went had a mesh fabric which also held the LCD monitor to show the code.

3). Shall contain hardware design element(s): We had a bunch of hardware components for our modernized jukebox. It contained the Arduino Uno, which acted as the "heart" of our project. We also had a Rock Pi 4B that was used for the RFID card scanner and voice commands. For user control, we also had three push buttons. There were also two limit switches to alert the stepper motor when it reached the end of the rail. The entire mechanical arm was moved up and down by a single stepper motor. We also had four servo motors that moved all of the bends in the mechanical arm. We also had servo and stepper motor drivers. In addition, RFID cards and a scanner were being used. The sound system also included two speakers and an amplifier. Furthermore, we displayed the music on Elizabeth’s computer and the Python code on the LCD screen which printed out the song that was

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

playing so the user could tell which song was playing. Finally, we used a microphone for voice commands.

4). Shall contain software design element(s): The RFID jukebox also contained software components. The Spotify API was one software component that was used to connect the RFID card to the song ID. PicoVoice was used for voice command control. Furthermore, all of the programs in the Arduino IDE were included in our project's software components. For example, the program for moving the mechanical arm's stepper and servo motors. The final software components were all of the Rock Pi's libraries and packages.

5). Shall apply knowledge from at least 2 previous courses (could be more than 2): at least one must be a core course, one course must be EE and one course must be CE: Because of our existing experience and knowledge from our classes, RFIDJs are able to satisfy requirement five. We have each taken courses that have provided us with sufficient knowledge to apply to our project. Each member of RFIDJs took either the electrical or computer engineering junior design courses. Ibrahima Diallo and Elizabeth Fatade worked with the Rock Pi in their computer engineering junior design course which was called Embedded and Mobile Systems. The information they learned from that course will be very useful for our project because we will be using the Rock Pi 4B for the scanning of the RFID cards and the voice command control. Gilberto Ruiz and Jenna Stapleton obtained experience with several microcontrollers in their junior design course called Digital Signal Processing and Controls Laboratory for electrical engineering. Working with multiple microcontrollers in that course will be useful with our project because we are using the Arduino Uno to control the servo and stepper motors and it is similar to the ones we have worked with previously. In addition, we all have prior experience in Python and C to work with the Rock Pi and Arduino. All the members of RFIDJs took the Introduction to Computing course in which we learned the basics of coding and learned how to code in Python. We also all took linux and C in the course Systems and Network Programming. We will be coding in C when working with the Arduino Uno and making the motors move in the Arduino IDE. We also have all taken the engineering ethics course, so we can apply what we have learned to our project.

6). Shall produce data that can be displayed in a graphical form: To satisfy requirement six, the RFIDJs produced and recorded data from our project and displayed it in a graphical form on our open house poster. RFIDJs recorded the accuracy of the voice command program. We tested it multiple times in different spaces. For example, we tested the program several times in a quiet area to see how accurate the program was. Then we tested the program again but in a loud area.

7). Shall have a user interface: For requirement seven, we had a few different user interfaces. The push buttons were part of the user interface because that was one of the ways the audience could use the jukebox. The voice recognition was also part of the user interface because the audience was able to tell the jukebox what to do. These were all of our user interfaces because the audience or user were able to interact with the product.

8). Shall identify relevant engineering standards that apply to your design: From the Engineering Ethics course we took, we were all familiar with and understand the engineering code of ethics. So, we were able to apply the engineering standards we learned from that course in our senior design project. We were able to make sure it was safe and wouldn't harm anyone by constructing an exterior shell that covered any cables, which is the first Engineering Code of Ethics rules of practice. Furthermore, because we are all senior computer and electrical engineers, we worked on a project that was within our area of expertise which was the second Code of Ethics rules of practice.

9). Shall have a demonstration that can be shown to an audience and judges, which complies with ECS Open House constraints: Our project satisfied all the open house constraints. The

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

outside shell was approximately 2 feet by 2 feet by 3 feet to fit all the hardware components in. We were able to transport it to the atrium with two team members so it fit the weight constraint. In addition, we had to detach some of the wires for it to be more easily carried to the atrium. The illumination in the atrium will have no impact on our project. Furthermore, our demonstration didn't last more than 5 minutes and the audience were very engaged. They had the option to let RFIDJs show them how our project worked or they could test out the jukebox themselves.

10). Shall produce applicable engineering drawings: To satisfy requirement ten, we included the software implementation flow chart in the proposed solution which is shown in Figure 3. The software implementation flow chart shows how the software will work together. In addition we have included a block diagram of all of our hardware components in Figure 1. The hardware block diagram depicts how all of the hardware components interact with one another. In addition we have pictorial representations of our physical components. In figure 6, we show the HDMI 7" 800x480 Display Backpack - Without Touch from Adafruit. In figure 5, it shows the Adafruit's Stereo 20W Class D Audio Amplifier - MAX9744. In figure 4, it shows the Visaton BG 17 Speaker. In figure 3, it shows the 13.56MHz RFID/NFC Card from Adafruit. In figure 2, the RFID Scanner is shown and it reads RFID cards with a 13.56MHz frequency. In figure 1, it shows last semester's progress of the Robotic arm moving with push buttons.

B. Appendix 2

1). What are the desired functions of the design: The desired function of our project is to play songs that have been assigned to individual RFID cards through push button manual control of a robotic claw or voice control.

2). What constraints related to the main function(s) must the design satisfy: Based on the Arduino Uno board we had limited open pins due to the board we chose for our project and the amount of other components that needed to be connected to the Arduino Uno. In addition, we used the Arduino Uno and Rock Pi because in our previous classes we used them and were familiar with them.

3). What are the performance objectives of the design? (Use quantitative metrics as much as possible): With our voice commands we wanted to create a sixty percent confidence level. We tried to get the voice recognition software to be as accurate as possible with different voices. To power every system of the jukebox, we required a steady source of 5.5V with a high current flow. To have the robotic claw going up and down each level accurately, the stepper motor needed to spin 1300 steps, we need this step to work one hundred percent of the time because any slight errors would mean that the jukebox would be unable to grab an RFID card as it wouldn't reach the appropriate level where the card is at.

4). What alternative design concepts were considered: In our analysis of alternatives we considered 4 different hardware swaps:

1. Adafruit CircuitPython MAX9744 Amplifier vs. TPA3118D2 Class-D Audio Amplifier
2. Arduino Uno vs Teensy vs Itsy Bitsy
3. Rock Pi 4B vs Raspberry Pi
4. RFID vs NFC

5). What analyses were used to select among the alternative design concepts: We used several different criteria in each of our analyses. Every criteria had a certain weight and we used a decision matrix to determine which choice was better for our project. In comparing the amplifiers we looked at max power output, input voltage range, efficiency, availability, cost, and channels and ended up choosing the MAX9744. In considering the I/O boards we looked at the microcontroller, operating voltage, clock speed, # of PWM pins, the current per PWM pin, familiarity, and compatibility. We ended up choosing the Arduino Uno. In choosing a single board computer we looked at the processor, memory,

RFID Jukebox Final Report

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

I/O options, storage, power, operating systems, price, familiarity, and availability. We ended up choosing the Rock Pi 4B. Finally, in choosing between using RFID card vs NFC cards we looked at purpose, range, data direction, and security. We ended up choosing to work with the RFID cards but acknowledged that the NFC cards were a very realistic option for our purposes.

6). Which concepts or skills learned in your previous coursework were applied to the design? (Please provide a list with each entry providing the course number of the course, plus a brief description of the concept or skill used.): We used much of our knowledge from previous classes for this final project. The junior design courses, ELE 392 and CSE 398 helped us prepare for designing and building a project on our own because we had a preview of it when designing and building our own junior design project. In the microcontrollers laboratory, ELE and CSE 397, we learned a great deal about different microcontrollers and the Arduino IDE. Experience with Linux and working from the terminal in CSE 384 helped us set up our Rock Pi's operating system, install packages/applications, and debug errors.

7). Evaluate your design, relative to its function(s) and constraints. How well did your design meet each of the performance objectives? How well does your design compare to other, existing solutions to the problem: We believe that our design met each of the performance objectives very well. We were able to get the jukebox to play music with push buttons and voice commands before the final deadline. During the open house, it worked perfectly for two hours, although it didn't run as smoothly during the final demonstration in class. We also were able to create our project with all the needed requirements. For example, the size, weight, and audience requirements were all fulfilled. In addition, although we didn't finish the joystick control, we planned to work around the constraints of not having enough pins for the push buttons on the joysticks. We were going to plan on using the push buttons with the joysticks.

When compared to existing jukebox implementations, our project provides a novel take on the music playing system. While sticking to the classic themes it is improved in the sense that you can map any song in the spotify library to an RFID card, as opposed to needing the specific record in the classic implementation.

When compared to the inspiration for our project, the RFID Jukebox offers a much better user experience. Instead of having to grab the card and scan it manually the user can use voice commands or the use of push buttons and watch the robotic arm select and play the song that they want. As one of the main goals of our product would be to entertain, this is a step up from existing solutions.

8). What impact do you see your design, if implemented, having upon public health, safety, and welfare, as well as upon current global, cultural, social, environmental, and economic concerns:

In terms of impact, our design could potentially harm the public health through physical damage due to the moving mechanical parts of the device, may alter a person's sleep schedule with the led's and monitor's blue light and it might cause discomfort to some people who would rather have a silent moment but couldn't due to the speakers being used. If this device is to be manufactured and fabricated at a worldwide scale, the impacts to the environment would be through contamination and air pollution. Our device will also need to be built to last as it will be built with precious computing materials such as gold, palladium, and platinum.

The economic impact from this device is the fact that in order to be able to utilize the RFID jukebox, the user must have a spotify premium account and synchronize it with the device. To have a Spotify premium account, they must buy it at a monthly rate. And lastly, due to our device having voice command capabilities, it may cause some impacts through privacy concerns of voice data being often stored by companies, discrimination from the

RFIDJs: Ibrahima Diallo, Elizabeth Fatade, Gilberto Ruiz, Jenna Stapleton
Syracuse University College of Engineering & Computer Science
ELE/CSE 492: Senior Design 2
Professor Graham

device not recognizing the user's commands because of his/her accent, and that the increase in use of voice commands in tech companies may result in job loss for individuals.

As an IoT device that collects audio from the user, it's important to handle this data responsibly to prevent abuse in the forms of identity theft, targeted advertising, stalking, etc. Beyond measures that we take to make sure that this data isn't being abused by the intended collector, it's important to not overlook security and IoT devices can be vulnerable to having this data taken forcibly.

9). What engineering standards were considered and why? If none were considered, explain why it was not necessary to consider engineering standards: One standard we considered throughout the project was "Engineers shall hold paramount the safety, health, and welfare of the public" because when designing and building this project we had to keep in mind the safety of the audience when using our product. We had to make an outer shell so no wires were poking out and the user couldn't accidentally touch anything they shouldn't. Another one we tried to keep in mind was "Engineers shall at all times strive to serve the public interest" because one of the requirements was to keep the audience engaged. So we had to keep thinking of ideas of how our demonstration could be more engaging for the audience.

10). Describe each student's role in the design project: Ibrahima worked with the RFID scanner, communication between our two boards, order forms, fitting hardware together, and troubleshooting issues on the Rock Pi
Elizabeth worked on making spotify API calls, handling voice recognition, and troubleshooting issues on the Rock Pi. Gilberto worked with the stepper motors and getting the arm to grab cards at each level, the amplifier and speakers, troubleshooting issues with the servo motors, fitted most of our hardware together and was very helpful in finishing the construction of the case. Jenna worked on getting the joysticks working with the servo motor arm and troubleshooting issues on our Arduino. Jenna also took up initiative and got the team going in the scheduling and almost all of our papers. Everybody took a role in communicating issues and lending help to troubleshoot anything, be it within their tasks or not.