Prototype of an Automated Wheelchair

Joanna Hughes   
School of Computer Science and Engineering  
University of the PacificStockton, California: [j\_hughes10@u.pacific.edu](mailto:j_hughes10@u.pacific.edu)

line 1: 2nd Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID  
  
line 1: 3rd Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID  
  
line 1: 4th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 5th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 6th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract*—Public service areas are a melting pot for a variety of potential patrons*.* These potential patrons will come with a wide variety of needs that call for specific solutions. One solution to a number of potential patron needs is access to automated systems. Specifically transportation through larger public areas from point A to point B. Smart Wheelchair Systems(SWS) help different patrons with different needs. In this paper we will explore an application of a SWS in a public service area (airport). We propose a prototype autonomous vehicle engineered from a smart car kit. Our prototype uses a microprocessor (Raspberry Pi 4) communicating with a microcontroller (Arduino) that will navigate to predetermined destinations. This application will potentially decrease the number of times human to human contact occurs in a public place and can help prevent the rapid spread of COVID-19 and other direct contact diseases.

Keywords— Arduino, Autonomous, IR sensors, Lavfin Robot Kit, Python, QR code, Raspberry Pi, Serial communication

# Introduction (*Heading 1*)

We have designed a prototype of an automated wheelchair. Our prototype can follow a guideline along the floor by using a Raspberry Pi with a camera module attached. Also, the camera reads QR codes to determine where it is located.

About 1.5% of the population uses wheelchairs and this amount may increase since there are people with temporary medical issues, in which they need a wheelchair or a walking cane. In 2018 over one billion people used airlines for travel and about three hundred and fifty-four million people requested assistance from an airline to move through the airport. Most of the time it is difficult for those people to find someone who can push their wheelchair. Therefore, they need to wait a long time at the airport for assistance.

Those people are very special for our research community, and we have decided to simplify their life by designing a  smart wheelchair prototype. The users are going to save time and money. Also, they will have the option to stop and go to any terminal at the airport, in which they are going to feel free since they are not going to have a person behind their back during their trip. Our users are going to feel more comfortable and safer.

# Lavfin Robot Kit And the Rpi

## Understanding the Arduino

Our team was working with a pre-designed robot or car that would be used to simulate the wheelchair. Once the robot was assembled, we began to study how a micro-controller can control a DC motor.  In our case, an Arduino UNO was used as a microcontroller and a L298N DC Motor Driver Module was used to control four DC motors that were intended to work independently.

Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), these pins provide an 8-bit PWM output. In our case, the digital pins have been configured as an output, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH , 0V for LOW. High is ON and low is OFF. Therefore, it is possible to send a signal to the DC driver and control the wheels, but first the following data was interpreted. The speed of a DC motor can be controlled by varying its input voltage. A common technique for doing this is to use PWM (Pulse Width Modulation). The average voltage is proportional to the width of the pulses known as Duty Cycle.

Figure 1 illustrates PWM technique with various duty cycles and average voltages.

Timeline

Description automatically generated

*Fig. 1 PWM techinique*

The higher the duty cycle, the greater the average voltage being applied to the dc motor (High Speed) and the lower the duty cycle, the less the average voltage being applied to the dc motor(Low Speed).

The DC motor’s spinning direction can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-Bridge. By using H Bridge we can run DC Motor in clockwise or anticlockwise directions. An H-Bridge is made up of four switches: two in series, and two in parallel, with the load placed in between the switches. In this configuration the circuit takes an “H” shape. Once this information was analyzed and interpreted, we were able to understand how a microcontroller can control one or more DC motors by using a DC motor driver. Therefore, we proceeded to understand the Arduino code that controls DC motors to comply with the engineering specifications. It was necessary to understand how the microcontroller can make the robot turn right, left, move forward or backward. We ended with the conclusion that For each of the L298N’s channels, there are two types of control pins which allow us to control speed and spinning direction of the DC motors at the same time.

*Fig 2. Image showing H-Bridge circuit used for controlling direction of rotation of a DC Motor*Shape, arrow

Description automatically generated

## RPI and Serial Communication

Another very important aspect was knowing how the user will be able to control the micro-controller and how to make the robot autonomous. We concluded that a micro-processor should be used. In this case, a microprocessor will make the micro-controller more intelligent. Therefore, a Raspberry PI was used as a microprocessor. It was determined that the best way to communicate a RPI with an Arduino is via serial communication. Serial communication is the most widely used approach to transfer information between data processing equipment and peripherals. It is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus.

Diagram

Description automatically generated

*Fig. 3. Serial communication diagram*

As seen in figure 3, for every clock pulse; the transmitter sends a single bit of data to the receiver. A python program was written to send one command at a time to the Arduino UNO.

## QR Codes

QR Codes are used as a reference for localization, in which the RPI camera scans it. A QR code is a scannable barcode encoded with data. Encoded means converted into a particular form. In the case of QR codes, numeric and alphanumeric characters, bytes, and kanji convert into a unique two-dimensional arrangement of squares. When an optical scanner passes over those squares, it translates their arrangement back into that data’s original form. The anatomy of a QR code is the following:

Qr code

Description automatically generated

*Fig. 4. QR code elements*

There are three position markers on every QR code. Consisting of an inner and outer eye, they allow scanners and cameras to quickly and accurately locate the data modules and the scanning direction (fig. 4).

## IR Sensors

The IR Line tracking sensor TCRT5000 module has an operation voltage between 3V and 5V that is perfect for all Arduino, ESP8266 and ESP32 microcontroller boards. The wavelength of the emitter LED is 950 nm that is on the top end of the infrared electromagnetic spectrum. The sensor has an IR light emitter and detector. The sensor returns the status of the IR light reflected from a surface as ON or OFF. The on-board LED shows the status of the sensor. Therefore, the sensor needs to be calibrated until the LED goes on or detects the black tape. The following graph shows the relative collector current in dependence of the operation distance.

*Graph 1. Relative Collector Current vs. Distance*

Chart, line chart

Description automatically generated

From graph 1, the peak operation distance of the TCRT5000 is 2.5 mm.

For safety features a HC-SR04 Ultrasonic sensor was taken into consideration and its data was interpreted. This sensor can be used as an object detector, in which the width of the received pulse is then used to calculate the distance to the reflected object as shown below in figure 5.

Diagram

Description automatically generated

*Fig. 5. Ultrasonic sensor*

From the picture shown above, a short ultrasonic pulse is transmitted at the time 0, reflected by an object. The sensor receives this signal and converts it to an electric signal. The next pulse can be transmitted when the echo has faded away. This time period is called the cycle period. The recommended cycle period should be no less than 50ms. If a 10μs width trigger pulse is sent to the signal pin, the Ultrasonic module will output eight 40kHz ultrasonic signals and detect the echo back. The measured distance is proportional to the echo pulse width and can be calculated by the formula above. If no obstacle is detected, the output pin will give a 38ms high level signal. The maximum range is 4 meters, and the minimum range is 2 centimeters.

# Methods And Results

## Supplies and Design

As restricted by guidelines in place for the Covid-19 Pandemic, our group conducted daily meetings and used the Lafvin Robot kit, a raspberry pi and a camera (Fig. 6) in order to make a prototype for the autonomous wheelchair on a small scale. To start we began by getting familiar with OpenCV, and image processing software, python and qr codes. After receiving the materials in the mail, we began by building the car as instructed in the kit (Fig. 7). The Lafvin smart robot came with an Arduino and pre-written code. The kit came with a Bluetooth module, which was plugged into the TX and RX pins. These pins are used in serial communication to receive and transmit data. In order to upload the code, those pins cannot be used because serial communication is used to upload to the Arduino and having both creates confusion and interference. The robot had four modes, line tracking, obstacle avoidance, infrared (IR) mode and Bluetooth mode. Using the IR remote to switch between modes proved to be difficult, although it was necessary to use it to check that each function of the car was working properly, and all wires were connected properly.

Letter

Description automatically generated with medium confidenceA picture containing indoor, yellow

Description automatically generated

Fig. 6 Supplies used for     Fig. 7  Original robot, built as by

the prototype        Lafvin instructions

After ensuring that the robot worked properly, we began setting up the Raspberry Pi 4 (RPI). This microprocessor will act as the “master” in the master slave connection between the Arduino. The Arduino and RPI communication through a serial connection. This can be achieved by either using the general-purpose input/output (GPIO) pins on the RPI and serial communication pins on the Arduino or through the Universal Serial Bus (USB) port on the RPI and the Universal Asynchronous Receiver/Transmitter (UART) port on the Arduino. Using the pins on both boards would, however, require a voltage shifter. To reduce cost and to avoid delaying our project, we used the USB connection. The chord that is used for uploading code simply plugs into the RPI instead.

Creating space for the additional hardware on the robot proved to be difficult. It is important to take into account that a wheelchair is considerably larger than our 12’’ long robots. Therefore, creating room for the power, Arduino, camera, and raspberry pi likely will not be as difficult. However, due to the length of wires and the camera chord, setting up the RPI without interfering with the previous design was nearly impossible. Each team member's robot had a unique set up. In addition to space restraints, the raspberry pi needed to be powered with 5 volts and 2 amps. Using an outlet to power the raspberry pi would create problems in its autonomy as well as a tripping hazard and decrease the distance it could drive. Two methods were taken in response to this issue. The simplest solution was to use a battery pack. An item many have at home to charge their mobile device. While this method was simple and effective, it increased the weight of the car. The other method, used by Ms. Hughes was to solder a RPI cord, the battery case and the cord to the Arduino to a voltage shifter, effectively powering the entire robot with two 3.7 volt lithium batteries.

## Turns

One securing the RPI to the car, we began our research on QR codes. After looking into the many types, we determined that the model one, the common QR code, would be the best option for our project. We began by writing a simple code to create and read QR codes in python. After completing that, we used online code generators for all future QR codes. Originally the QR codes would contain one letter, this letter would be sent to the Arduino through the serial connection. The Arduino code has a function for serial control that runs the motors when it picks up a letter from the serial connection. By creating QR codes with letters, the camera would scan the code effectively controlling and driving the robot. At the beginning of the project, we used the Arduino code that came with the Lavfin kit, however it contained more functions than needed and in order to modify the code to what was needed many things were deleted and changed. The main commands were left, right, forward, back and line tracking. We combined these functions to create another command, called by the RPI with letter ‘K.” The function is named “seek line” and simply drives the car forward until reaching a black line, at which point the car stops. This function was essential in making turns or continuing past the QR code. When the car sees the QR code it immediately stops at the white border. For a forward command, the code turns line tracking off, drives the car until it finds the black line, and begins line tracking until it finds the next QR code. Turning is a similar process. The camera reads an “L” or an “R,” and turns that direction for half of a second. It then advances forward until finding the line and tracking it to the next code. This essentially creates a wide turn.

National Science Foundation

## Navigating the Map

## Once the code was written to read the codes and make turns, we began making simple if else statements and taking user input to navigate. At this point the group needed a way to collaborate on code virtually. For this we used replit.com. Once running the program, it would ask the user, “Left or Right.” After successful runs using that code on the RPI, the next step was to make a map. The first step was to create a map with multiple terminals and gates. At each gate and intersection there was a QR code labeled with a unique individual number (fig. 8). Using python dictionaries, we created a map. This dictionary contained three items: the destination, the number of the local QR code, and which direction to go from there. Furthermore, the code asked the user which terminal and gate they wanted to go to. After, “Have a safe trip!” was printed to the screen and the robot regan following the line(fig.9). To explain this further, imagine the user wanted to go to terminal 2 gate B. Refer to figure three to follow along. The robot will drive up to QR code 1, send T1GB and 1 to the function which finds the corresponding direction to the next QR code in the map dictionary. In this case the function returns an “R.” The car takes a right turn and follows the line to code 9. At this point it repeats the process, the code sending back an “L.” This continues on until the car arrives at the Gate the user imputed. In this example, at code 7, Terminal 2 gate B, the camera reads in this code and runs the arrival function. This stops the car, and prints out to the user “You have arrived.” Thus navigating the prototype through the small scale airport using black tape and QR codes.

Graphical user interface, text, application, chat or text message

Description automatically generated

*Fig. 8 Python code creating map*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Fig. 9 Map navigated by the prototype*

# Conclutions and Future Work

After many new experiences, failures and successes, our team was successfully able to create an autonomous robot that could navigate to a terminal from the start by taking user input, information from QR codes and black lines. For future development or to reproduce the project, it is important to consider the following recommendations.

When using the IR sensors to detect lines, the width of the tape or paint can affect the tracking. For sensors of that size, the tape needs to be around an inch in width, or for wider tape the sensors need to be spread apart. Any smaller line will likely be ignored by the sensors. Additionally, the Arduino code in our final version did not include the ability to navigate and use the IR sensing to detect and avoid obstacles. Scanning the area and reading the information while line tracking slowed the transmission of the line sensors, which made the car lose the line constantly. Finding a way to read the information to the sensors and move the motors in a timely manner would need many more trials and more research.

For power, the most effective, though energy consuming, was using the voltage shifter in order to have the entire car run on one power source. However, a different type or larger battery will be needed for the actual wheelchair, or any long use of the prototype. The lithium-ion batteries that came in the kit were rechargeable, however no way to charge these batteries was included. Our group had to each buy a charger, however, switching to a more common battery type may solve the problem with the battery running low quickly as well.

Buying a charger for the batteries was unfortunately, only one of the downfalls to the Lavfin kit. While the pre-written code that came with the kit likely saved us time, made it difficult to understand. Writing our own code would have helped us understand the functions and commands the car needed written in the Arduino code. It also would have been a great skill to work on and learn. Additionally, there were some features that were not needed in the final project, such as the IR sensors. A new version of this car could use a simpler base kit. Furthermore, in future models having the wheels firmly attached with some screws, as well as additional screws to anchor down the motors. This would reduce or even eliminate the issues with the car not being able to follow a straight path. Finally, a simple suggestion in regard to wiring the motors. Stripping the wires, twisting the ends, and soldering them to the board will save from a dislodged wire and malfunctioning motors in the future.

Our group had a difficult time undergoing this project remotely. There were often many miscommunications and confusions, but the group was always willing to communicate, talk and set up meetings. Each member contributed more at certain portions of the project. Jeremiah understood the maps and researched other options for our navigation such as the Json or Dijkstra algorithm. These ideas were not implemented in the prototype but helped us gain the information and ideas needed to create the maps. Such algorithms and ideas would be very beneficial in future versions. The change Cesar Valdez made to the Arduino code to make the car stop at a QR code, read it and drive past was instrumental in making turns and navigating. The method of turning and the final python code was created by Joanna Hughes. Through much research and communication our team was able to lean on and learn from each other, helping us reach the goal of navigating from start to a terminal gate.

In the future, a web app could be made to create a simpler user interface, as well as having the car immediately start the program upon powering on.

##### Acknowledgment

Thank you to Dr. Kofi Nyarko, Professor Hashmath Fatima.

Mr. Andre Lewis, Mr. Kelechi and Nile Walker and the Morgan State University Staff that helped guide us through this project. Special thanks to Shayne Hughes for all the explanations and additional support.

##### References

[1]AdrieSentosa and Instructables, “Raspberry Pi - Arduino Serial Communication,” *Instructables*, 03-Oct-2017. [Online]. Available: https://www.instructables.com/Raspberry-Pi-Arduino-Serial-Communication/. [Accessed: 25-Jul-2021].

[2] J. Barnett, “Controlling DC Motors Using Python With a Raspberry Pi,” *Business Envato Tuts+*, 18-Mar-2021. [Online]. Available: https://business.tutsplus.com/tutorials/controlling-dc-motors-using-python-with-a-raspberry-pi--cms-20051. [Accessed: 25-Jul-2021].

[3] M. M. Waldrop, “Autonomous vehicles: No drivers required,” *Nature News*, 04-Feb-2015. [Online]. Available: https://www.nature.com/articles/518020a. [Accessed: 25-Jul-2021].

[4] “Physical Computing with Python,” *projects.raspberrypi.org*. [Online]. Available: https://projects.raspberrypi.org/en/projects/physical-computing/14. [Accessed: 25-Jul-2021].

[5] “Raspberry Pi Arduino Serial Communication - Everything You Need To Know - The Robotics Back,” *End*, 18-Aug-2020. [Online]. Available: https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/. [Accessed: 25-Jul-2021].

[6] S. Dholakia, “How Autonomous Vehicles Work,” *Medium*, 12-Mar-2019. [Online]. Available: https://swarit.medium.com/how-autonomous-vehicles-work-1a54364463c6. [Accessed: 25-Jul-2021].

[7] “What are possible ways to communicate between two microcontrollers?,” *projectsflix*, 09-Nov-2020. [Online]. Available: https://projectsflix.com/raspberry-pi/what-are-possible-methods-to-communicate-two-microcontrollers/. [Accessed: 25-Jul-2021].

[8] “What is an Autonomous Car? – How Self-Driving CarsWork,” *Synopsys*. [Online]. Available: https://www.synopsys.com/automotive/what-is-autonomous-car.html. [Accessed: 25-Jul-2021].