11-13-2023

The Automated wheelchair

Jonathan Doksa Kamba Ngoyi

Automated wheelchair

An honors Thesis submitted in partial fulfillment of the requirements for Honors in Electrical Engineering

By

Jonathan Doksa Kamba Ngoyi

Under the mentorship of Dr. Fernando Rios-Gutierrez

ABSTRACT

Automation refers to the use of technology to execute tasks through programmed instructions and automated control mechanisms to ensure accurate execution. This minimizes human involvement and encompasses various fields, including business process automation. A notable form of automation is assistive technology, which refers to any device or software that enhances individuals' learning, communication, or overall functionality. This can range from advanced tools like computers to simpler aids like walking sticks. Assistive technology aims to promote inclusivity and participation, particularly among individuals with disabilities, the elderly, and those with non-communicable diseases.

The Automated Airport Wheelchair is a culmination of prior projects developed over time. This wheelchair offers two modes of operation: automatic and manual. In the automatic mode, the wheelchair can either go to a desired location set in the wheelchair or follow a friend via an app.

Thesis Mentor: Dr. Fernando Rios-Gutierrez

Honors Dean: Dr. Steven Engel

November 2023
Department of Electrical Engineering
Honors College
Georgia Southern University

Acknowledgments

I want to start by expressing my sincere gratitude to my research mentor, Dr. Fernando Rios-Gutierrez, for his unwavering support and advice during my undergraduate research career. This thesis project would not have been possible without the invaluable mentoring he provided. It has been a pleasure to collaborate with him both this semester and in the past, and I am appreciative of the information and experience our study has given me. I am particularly appreciative of the Electrical Engineering
Department for providing possibilities for undergraduate engineering students to take part in research projects. In addition, I want to thank the Honors College for inspiring students to participate in initiatives like this one. Finally, I want to express my gratitude to my parents for enabling me to pursue a career in Electrical engineering at such a great school as Georgia Southern University in which I learned valuable lessons that will help me shape my future the way I see it.

CONTENT:

INTRODUCTION……....4

LITERATURE REVIEW……7

METHODS……...9

Building the wheelchair……...9

Material…….....11

Testing Components……..14

RESULTS……....23

DISCUSSION AND CONCLUSION……....43

FUTURE WORK……44

REFERENCES……44

INTRODUCTION:

Wheelchairs have a long and storied history, evolving from rudimentary devices into sophisticated aids for those with mobility challenges. The concept of a wheelchair dates back centuries, with early designs attributed to ancient Chinese ingenuity, as far back as the 6th century. This early rendition featured a rather rudimentary wooden platform made with a set of wheels, offering rudimentary mobility for individuals with limited leg function. Nevertheless, these initial wheelchairs suffered from a lack of user-friendliness and notably lacked any form of automation. Thus, they relied heavily on physical exertion, proving to be unwieldy and far from convenient, resembling little more than wooden platforms mounted on wheels. It wasn't until the late 19th century that wheelchairs saw significant improvement, with the development of collapsible versions for easy transportation. These early innovations paved the way for modern motorized wheelchairs and advanced mobility solutions.

The official early signs of automated wheelchairs similar to what we know nowadays was during the 20th century. Compact electric motors, improved battery technologies, and the use of advanced materials paved the way for the development of more practical and efficient designs. The integration of sensors, computers, and innovative control mechanisms significantly improved maneuverability and safety. Early electronic wheelchairs were controlled through basic switches and joysticks, laying the foundation for the sophisticated control systems that have since come into existence.

The field of automated wheelchair control technology has witnessed substantial progress throughout time, marked by enhancements in both hardware and software components. Nevertheless, it is important to acknowledge that various limitations and difficulties persist. An example would be from operating these devices within unsafe surroundings. Conventional joystick-based controls demand extensive training and proficiency, thus, posing a challenge in

scenarios that demand precise and immediate responses. Furthermore, joystick controls may lack the necessary intuitiveness and responsiveness for specific applications.

Arduino Uno, a versatile microcontroller board, has been widely embraced within the realm of assistive technology. Its accessible open-source nature and user-friendly programming capabilities render it an ideal choice for crafting tailored solutions to cater to the distinctive requirements of individuals with disabilities. Arduino Uno has been effectively employed to conceive inventive devices that enrich the daily lives of those grappling with mobility constraints or communication impediments. Ranging from systems governing wheelchair mobility to communication aids and environment control mechanisms, the remarkable adaptability of Arduino Uno has assumed a central role in making assistive technology more inclusive and accessible.

Applications, often referred to as "apps," have also wielded a profound influence within the realm of assistive technology. Their impact has been truly transformational, significantly enhancing the quality of life for individuals contending with disabilities. These remarkable digital tools offer a wide range of solutions, spanning from aiding non-verbal individuals in communication to providing essential navigation assistance for those with visual impairments. The app created for this project was made using the MIT App creator. This website is a valuable resource for individuals looking to enter the world of app development without an extensive programming background. It simplifies the app creation process, making it accessible for a broad audience.

In this thesis, we will examine the use of MIT App creator in improving conventional automated wheelchair techniques. To make this possible, an Arduino, Bluetooth module, a GPS and a compass will be connected together with a breadboard so that the wheelchair knows where it is

and where it is going at once. Using the features available in the MIT app creator, an app will be created capable of tracking the GPS location of the phone (with the app) and send it to the Arduino continuously. Similarly, the wheelchair will be able to track its own location then drive to the phone's location to close in the gap between the two. This application can help the wheelchair in following a particular subject in the airport or even go to particular gates if needed.

This paper is organized as follows: The method will explain every step in the creation of the app, the testing of the electronics and the connection between all electronic components and Arduino. The results section will present the results we got from the methods we used for this project, and then there will be a discussion and conclusion section to give the conclusions we made based on the results we got during the experiment.

LITERATURE REVIEW:

The use of Apps and automation techniques has recently been used to control various types of assistive devices. This literature review provides insight into the current research regarding assistive technologies.

In a first study, Aaron et al.(2014) used a similar approach to make a device follow a particular user called the “Follow me cooler”. The cooler in this project could follow the user with refrigerated drinks and go to specific locations as well. The design includes a configuration featuring two motorized front wheels along with a single swivel wheel at the rear. In terms of its electronic components, the cooler is equipped with a GPS and Bluetooth module, a compass, an Arduino, and an additional Bluetooth module as its primary constituents. Activation of the automated mode is facilitated through the utilization of the 'Blink' app, establishing a wireless connection between the smartphone and the microcontroller. Remarkably, in this automated mode, the project offers the capability to autonomously trail the user's movements and even navigate to predefined locations as specified by the user. Furthermore, the cooler is furnished with a motor that seamlessly opens the lid when required, eliminating the necessity for manual intervention in this task.

Another study by Zannatul Raiyan et al. (2018) showcased the design of an Arduino based voice controlled automated wheelchair. In this design, a voice recognition system allowed the physically disabled person to control the wheelchair by voice command who have issues in hand movement due to ageing or paralysis for joystick-controlled wheelchairs. Additional features such as obstacle detection for the safe movement and a GSM based navigation system for tracking and sending notifications to increase the usability of the automated wheelchair system were also added to the overall design. The implementation of the design was made

using an Arduino Mega2560, Easy VR3 speech recognition module, SIM900A GSM module and relay-based motor controller circuits.

Overall, the studies reviewed showed that Apps and Arduino Uno can be used to control automated devices to accomplish specific tasks which can be translated to assistive technology. However, the complexity of these tasks and the accuracy of the results (avoiding multiple obstacles, how to react in steep roads for example) greatly depend on how complex the code for Arduino is. Ultimately, more electronic components, sensors, and research would be needed to make these types of automated devices ready for any situations.

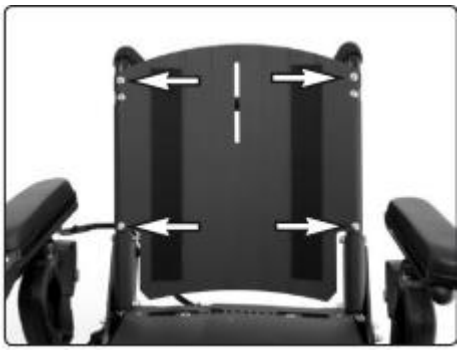
METHODS:

I- Building the wheelchair

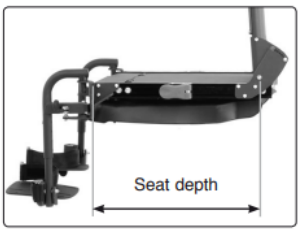
1.1- Wheelchair frame

Figure 1: Permobil M300 PS Junior Power Wheelchair Base.

For this project, the wheelchair will be built using the Permobil M300 PS Junior Power Wheelchair Base (figure 1). This kit is designed to offer a high level of flexibility and customization, making it easier to adapt to the user's specific needs. With the tires, screws and other components already included, it should make the project's implementation much easier.



Seat depth	Max. user weight
10"	66 lbs
11"	77 lbs
12"	88 lbs
13"	99 lbs
14"	121 lbs
15"	143 lbs
16"	154 lbs
17"	165 lbs
18"	165 lbs



Seat depth setting.

Figure 2: Adjusting the seat width and depth

Upon receiving the kit, we would need to securely attach the base to the main frame according to the manufacturer's instructions, making sure it's properly aligned and fastened. It is also important to note that settings such as such as seat width, seat depth are also adjustable in this case, leaving it open for many possibilities. For this project we will adjust it to standard values, with seat width to 20 inches and depth at 18 inches. After it is done, we can move forward with the seat assembly by customizing other seating system according to the user's needs. This typically includes attaching the seat and backrest to the frame. Secure these components as per the kit's instructions.

Figure 3: Adjusting arm and foot rest

With these steps completed, we can Attach the armrests and footrests (figure 3) to the frame at the designated locations on either side of the seat. For each it is important that they accommodate to different legs and arm’s lengths and angles (for leg rest) so that many users can have access to it.

1.2- Wheelchair electronic compartment

Now that the wheelchair has been integrated, the subsequent phase involves crafting the enclosure that will house all of the wheelchair's electronic components. In this project, the enclosure will be fabricated through 3D printing and designed using computer-aided design (CAD) software. Tinkercad will be employed for its simplicity and user-friendly interface. The design showcased in Figure 4 was chosen due to its compact height, ensuring the safety of the electronics without any risk of contact with the ground. Furthermore, two openings have been integrated beneath the design to facilitate ventilation and enable the dissipation of heat generated by the electronic components.

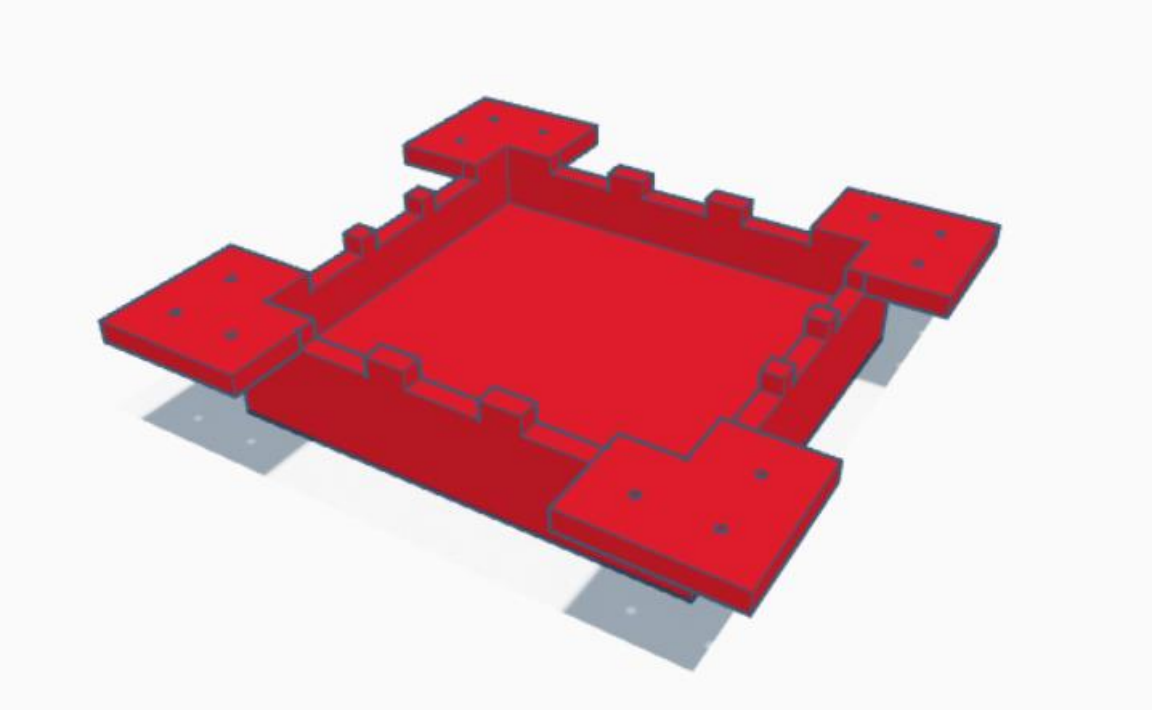


Figure 4: Electronic compartment

II- Materials

2.1- HC-05 Bluetooth Module

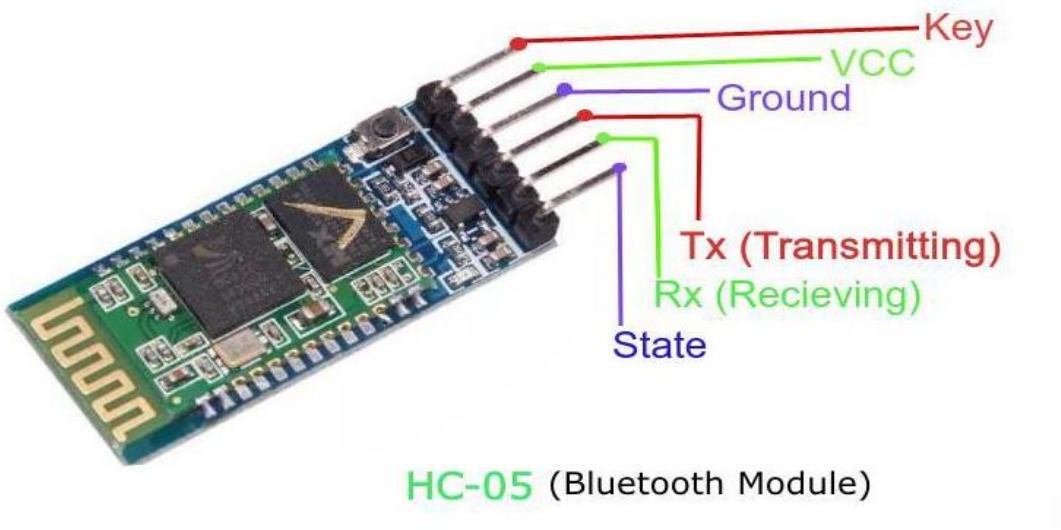


Figure 5: Bluetooth Module

The HC-05, a class 2 Bluetooth module, is crafted for seamless wireless serial communication. It comes pre-configured as a slave Bluetooth device. Once paired with a master Bluetooth device like a PC, smartphone, or tablet, it operates transparently for the user. Any data received through the serial input is instantly transmitted wirelessly. When the module intercepts wireless data, it's relayed through the serial interface without requiring any user-specific code in the microcontroller program.

2.2- HMC5883L Compass Sensor

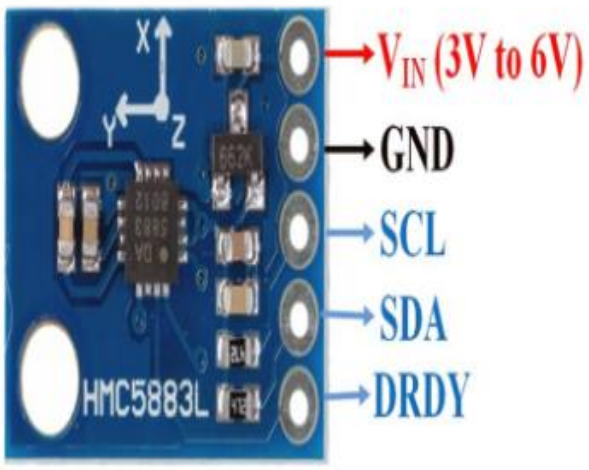


Figure 6: HMC5883L Compass

HMC5883L is a 3-axis digital compass used for two general purposes: to measure the magnetization of a magnetic material like a ferromagnet, or to measure the strength and, in some cases, the direction of the magnetic field at a point in space.

2.3- GPS Module

Figure 7: Parallax PAM-7Q GPS Module

The Parallax PAM-7Q GPS Module is a compact and versatile GPS module that provides highly accurate positioning and navigation capabilities. It communicates with GPS, GLONASS, and

QZSS satellite systems, offering multi-constellation support for enhanced accuracy and reliability. Designed for integration with microcontrollers and other hardware, it's suitable for applications like vehicle tracking, drone navigation, and IoT devices. With low power consumption and an onboard antenna, this module simplifies the setup and is ideal for various projects requiring precise geospatial information.

2.4- Golden Motor HPM3000B

Figure 8: Wheelchair's motor

The Golden Motor HPM3000B is a high-performance brushless DC (BLDC) electric motor manufactured by Golden Motor. With a potential power output of 3000 watts, this motor is designed for applications requiring substantial torque and speed. Following the typical characteristics of BLDC motors, it boasts a brushless design, ensuring efficiency, durability, and precise control with minimal maintenance needs. When coupled with its own motor driver that comes with the kit or a Sabertooth Dual Motor Driver, It communicates with the Arduino through simple serial commands, making it relatively straightforward to integrate into Arduino-based projects.

2.5- Turnigy Graphene 6S 5000mAh 22.2V LiPo Battery

Figure 9: Lipo Battery

The Turnigy Graphene 6S 5000mAh 22.2V LiPo Battery is a high-performance lithium polymer battery designed for various applications, including electric vehicles and hobbyist use. Known for its advanced graphene composite technology, this battery offers a remarkable combination of high energy density, reduced weight, and improved discharge capabilities. The 6S configuration indicates six cells in series, providing a nominal voltage of 22.2V. With a capacity of 5000mAh, it delivers a substantial amount of power, making it suitable for demanding applications such as electric mobility devices, drones, and RC vehicles.

III- Testing the Components

3.1- Testing the Bluetooth Module

To verify the functionality of the Bluetooth module and ensure its compatibility with other Bluetooth devices, we followed a simple procedure. We connected the power and ground terminals to the Bluetooth module and attached a red LED to the "State" pin. Subsequently, we initiated a wireless connection between the device and the Bluetooth module. When a successful Bluetooth connection is established with the module, the voltage of the "State" pin transitions from LOW to

HIGH. This transition causes the LED to illuminate, indicating the correct connection of the Bluetooth module.

(It's important to note that the HC-05 Bluetooth module has limited compatibility with certain devices. For instance, devices like iPhones may not establish a connection with the Bluetooth components due to differences in connection parameters.)

3.2- Testing the GPS Module

To test the GPS Module, we used a sample test code for that component. This test code sets an Rx and Tx pin connected to the Arduino. This code also includes the necessary libraries to initialize a serial connection between the GPS module and the Arduino and to initialize the GPS module as an object in the code. After running the code and allowing the device to connect to satellite position information, it would display a lot of information coming from the GPS into the Arduino serial monitor. The test code is as follows:

```
#include "TinyGPS++.h"
#include "SoftwareSerial.h"

SoftwareSerial serial_connection(6,13); //RX=pin 10, TX= pin 11
TinyGPSPlus gps; //GPS object

void setup()
{
  Serial.begin(9600);
```

```
  serial_connection.begin(9600);
  Serial.println("GPS Start");
}

void loop()
{
  while(serial_connection.available())
  {
    gps.encode(serial_connection.read());
  }
  if(gps.location.isUpdated())
  {
    Serial.println("Latitude:");
    Serial.println(gps.location.lat(),6);
    Serial.println("Longitude:");
    Serial.println(gps.location.lng(),6);
    Serial.println("Altitude(ft):");
    Serial.println(gps.altitude.feet());
    Serial.println("Speed(MPH):");
    Serial.println(gps.speed.mph(),6);
```

```
    Serial.println("");
  }
}
```

This code would accurately display the GPS latitude position, longitude position, speed, and altitude. With this being completed, we knew the GPS module was working as intended and was not faulty.

3.3- Testing the compass

The same process from testing the GPS module is applied when testing the compass. Set an Rx and Tx pin, connect those pins to the compass, and run a sample test code. There were multiple test codes for this device, but we only needed to test one of them. The code for testing the compass is as follows:

```
#include <QMC5883LCompass.h>

QMC5883LCompass compass;

void setup() {
  Serial.begin(9600);
  compass.init();
```

```
}

void loop() {
  int x, y, z;

  // Read compass values
  compass.read();

  // Return XYZ readings
  x = compass.getX();
  y = compass.getY();
  z = compass.getZ();

  Serial.print("X: ");
  Serial.print(x);
  Serial.print(" Y: ");
  Serial.print(y);
  Serial.print(" Z: ");
  Serial.print(z);
```

```
  Serial.println();


  delay(250);
}
```

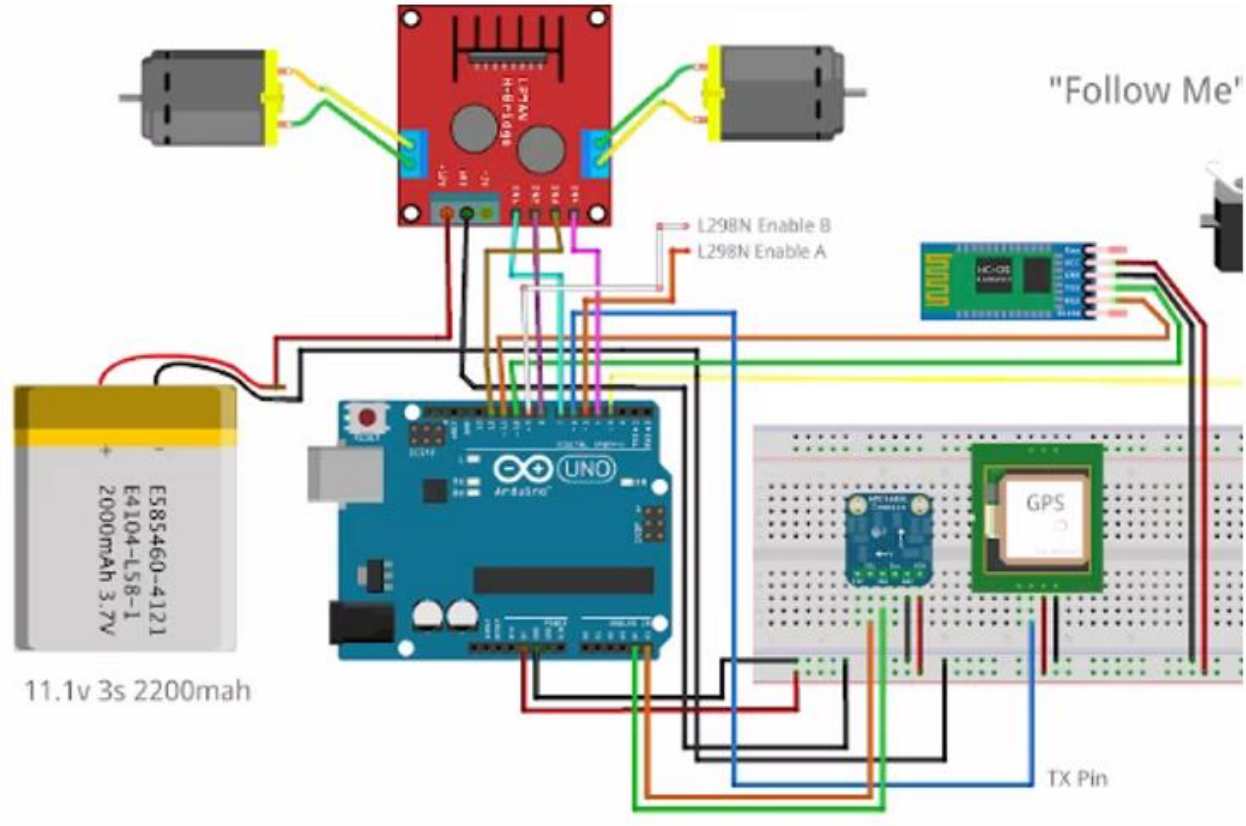


Figure 10: Electronic components connected together

Once all the electronic components are assembled and have undergone successful testing, we will utilize the schematic shown in Figure 10 to establish their interconnections within the electronic compartment. This schematic draws inspiration from the "Follow Me Cooler" project

and will empower the wheelchair to autonomously follow the user by triggering and orchestrating the electronic components to function in unison as originally envisioned.

3.4- Creating the tracking App

For the Arduino to understand the GPS location of the and subsequently know where to go, an app was created that could connect with Arduino through the Bluetooth module. This app was created using the MIT app inventor. The MIT's App Inventor is a web-based platform for creating Android applications. In this platform, after creating the new project, we can then design the user interface by adding various buttons and labels for controlling and displaying the GPS data. Once it is done, we can then add the location sensor to the overall design by going to the designer view, under the “Palette” option then looking for “LocationSensor” component. With the location sensor selected, we can now set its properties, specifying the desired accuracy and interval for receiving location updates. With the sensor in action, it's time to set up event handlers in the Blocks view to effectively respond to location changes, ensuring a seamless experience for users.

We can now use the blocks in the software to setup location updates. In the Blocks view, we will set up event handlers for location updates, by going under the “Built-in” layer and selecting the blocks related to the LocationSensor part. The "LocationSensor1.LocationChanged" block will be the one used to handle location updates. Additionally, the Label blocks will be used to display the latitude and longitude data from the app. With that done we can now connect the Android device to the computer, use the connect button to connect the app to the device then test the app on the connected device.

3.5- Sending GPS coordinates to Arduino

With the Bluetooth module connected to the Arduino as described, we can then create an event handler to send the GPS data to the Bluetooth module. This can be done by selecting the BluetoothClient component in MIT App Inventor to establish a connection with the paired Bluetooth module. We would also need to specify the module's Bluetooth address in the app's design. The GPS coordinates will be then sent as as strings via Bluetooth to the connected Bluetooth module. This can be done using the "BluetoothClient" component's "SendText" method or similar functions. Once it is done the string received can be used in the code to initiate a response from the wheelchair.

3.6- Wheelchair follow code

The very first step in the code will be to import all necessary libraries using the imports keyword. With that, functionalities for GPS, Bluetooth communication, motor control, and a compass will be incorporated so they can work together. One thing to note is that the code used for this project is heavily inspired from the “Follow me cooler” made previously. This is mainly due to the similarities between the functions of the cooler and that of the wheelchair that is designed in this project. The code then initializes a TinyGPS++ object for parsing GPS data. It includes functions for checking GPS coordinates from a GPS module and converting the data into latitude and longitude. As for the Bluetooth communication, it sets up a SoftwareSerial object to communicate with our android and receive the GPS coordinates. Additionally, it initializes a compass module (QMC5883L) and includes functions for reading compass data, calculating headings, and adjusting for offset and declination. It then provides functions for controlling the motors connected to the Arduino and functions for driving to a specific GPS location, with adjustments for distance, direction, and steering. Using the drive to() function incorporated in the code, the wheelchair will be able to

adjusts its movement based on the difference between its current position and the target GPS coordinate.

RESULTS:

The research led to a successful mobile app (figure 1 & 2) that sends a user's location to an Arduino system. We followed instructions for app development and connected the app to the hardware. We also made sure all electronic parts worked together correctly by following the model from the "follow me cooler" presented previously. Through testing, we confirmed that the compass, GPS, and motor controller all worked as expected. This shows that our system is ready for practical use.

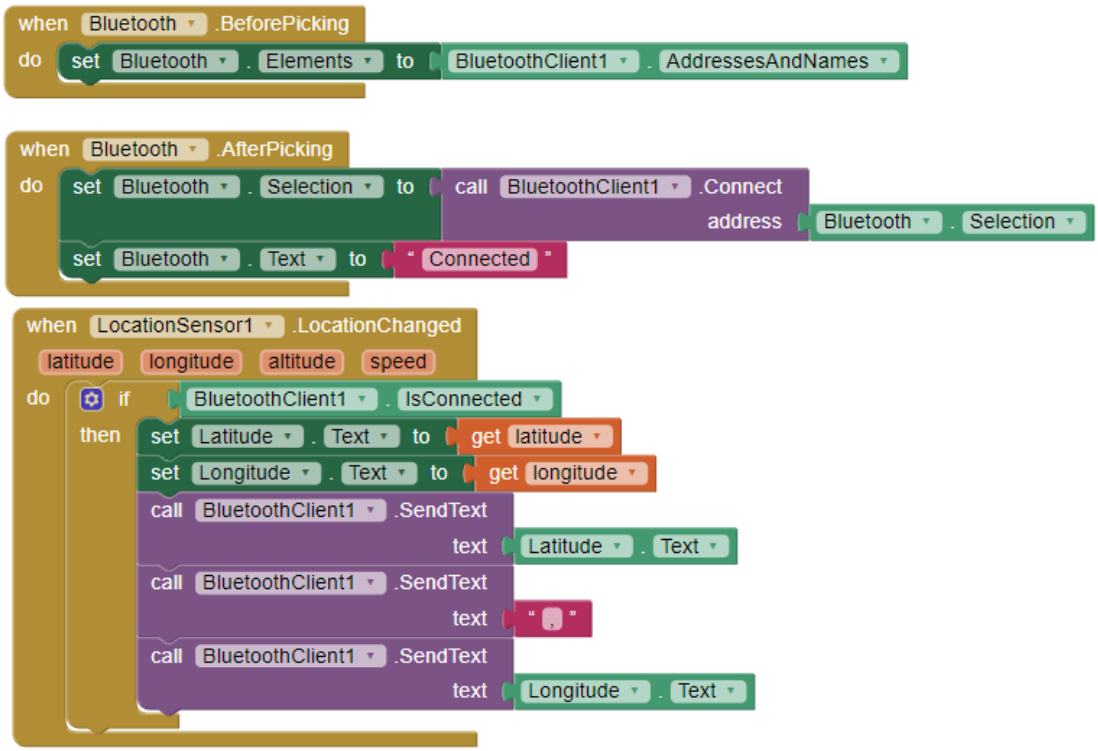


Figure 11: App Block Code

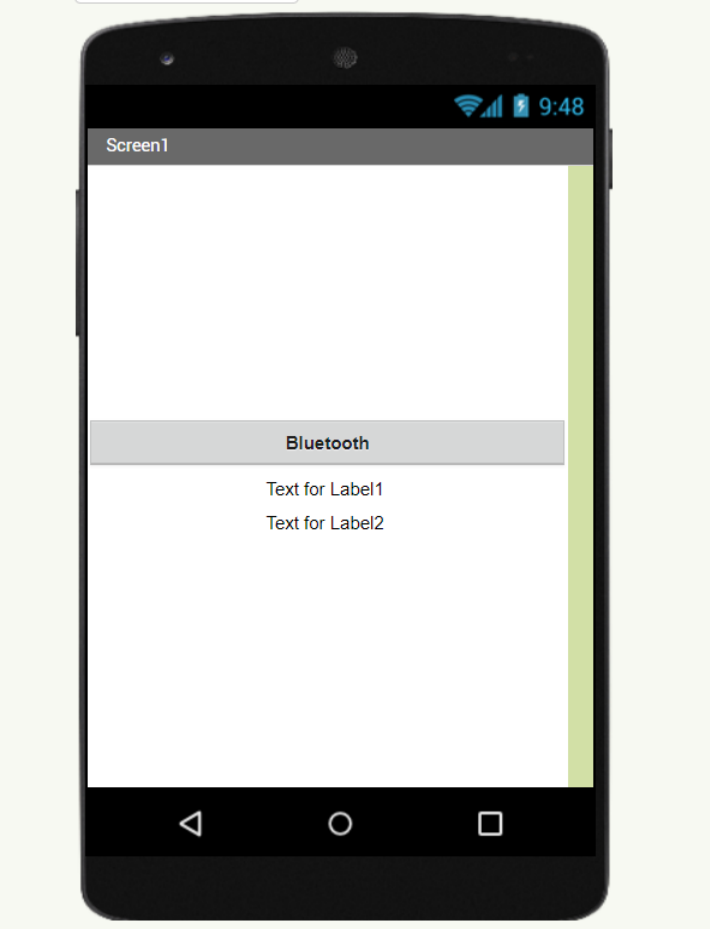


Figure 12: Phone screen Creation

Figure 13: Working code

```
// Imports

#include <Wire.h>

#include <QMC5883LCompass.h>

#include <Servo.h>

#include <SoftwareSerial.h>

#include "TinyGPS++.h"                // Use local version of this library

#include "./WagonDefinitions.h"


// GPS

TinyGPSPlus gps;


// Master Enable

bool enabled = false;
```

```
//WidgetTerminal terminal(V3);

// Serial components
SoftwareSerial bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);
SoftwareSerial nss(GPS_TX_PIN, GPS_RX_PIN);           // TXD to digital pin 6

/* Compass */
QMC5883LCompass compass;

GeoLoc checkGPS()
{
  GeoLoc wagonLoc;

  float x = checkGPSX();
  delay(200);
  float y = checkGPSY();

  wagonLoc.Lat = x;
  wagonLoc.Lng = y;

  return wagonLoc;
}

float checkGPSX()
```

```
{
  float x;
  nss.listen();
  while(nss.available())
  {
    gps.encode(nss.read());
  }
  if(gps.location.isUpdated())
  {
    x = gps.location.lat();

    return x;
  }

  return x;

}

float checkGPSY()
{
  float y;
  nss.listen();
  while(nss.available())
  {
```

```
    gps.encode(nss.read());
  }
  if(gps.location.isUpdated())
  {
    y = gps.location.lng();

    return y;
  }

  return y;

}

GeoLoc checkPhone()
{
  GeoLoc phoneLoc;

  float x = checkPhoneX();
  delay(200);
  float y = checkPhoneY();

  phoneLoc.Lat = x;
  phoneLoc.Lng = y;
```

```
  return phoneLoc;
}

float checkPhoneX()
{
  String Phone;
  float x,y;
  bluetoothSerial.listen();
  while(bluetoothSerial.available())
  {
    Phone = bluetoothSerial.readString();
    for (int i = 0; i < Phone.length(); i++)
    {
      if (Phone.substring(i, i+1) == ",")
      {
        x = Phone.substring(0,i).toFloat();
        y = Phone.substring(i+1).toFloat();
      }
    }
    return x;
  }

  return x;
}
```

```
float checkPhoneY()
{
  String Phone;
  float x, y;
  bluetoothSerial.listen();
  while(bluetoothSerial.available())
  {
    Phone = bluetoothSerial.readString();
    for (int i = 0; i < Phone.length(); i++)
    {
      if (Phone.substring(i, i+1) == ",")
      {
        x = Phone.substring(0,i).toFloat();
        y = Phone.substring(i+1).toFloat();
      }
    }
    return y;
  }

  return y;
}

#ifndef DEGTORAD
```

```
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.295779513082320876f
#endif

float geoBearing(struct GeoLoc &a, struct GeoLoc &b)
{
  float y = sin(b.Lng-a.Lng) * cos(b.Lat);
  float x = cos(a.Lat)*sin(b.Lat) - sin(a.Lat)*cos(b.Lat)*cos(b.Lng-a.Lng);
  return atan2(y, x) * RADTODEG;
}

float geoDistance(struct GeoLoc &a, struct GeoLoc &b)
{
  const float R = 6371000; // km
  float p1 = a.Lat * DEGTORAD;
  float p2 = b.Lat * DEGTORAD;
  float dp = (b.Lat-a.Lat) * DEGTORAD;
  float dl = (b.Lng-a.Lng) * DEGTORAD;

  float x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));

  return R * y;
}
```

```
float geoHeading()
{
  int x, y;
  compass.read();

  x = compass.getX();
  y = compass.getY();

  // Hold the module so that Z is pointing 'up' and you can measure the heading with x&y
  // Calculate heading when the magnetometer is level, then correct for signs of axis.
  float heading = atan2(y, x);

  // Offset
  heading -= DECLINATION_ANGLE;
  heading -= COMPASS_OFFSET;

  // Correct for when signs are reversed.
  if(heading < 0)
    heading += 2*PI;

  // Check for wrap due to addition of declination.
  if(heading > 2*PI)
    heading -= 2*PI;
```

```
  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI;

  // Map to -180 - 180
  while (headingDegrees < -180) headingDegrees += 360;
  while (headingDegrees >  180) headingDegrees -= 360;

  return headingDegrees;
}

void setSpeedMotorA(int speed)
{
  digitalWrite(MOTOR_A_IN_1_PIN, LOW);
  digitalWrite(MOTOR_A_IN_2_PIN, HIGH);

  // set speed to 200 out of possible range 0~255
  analogWrite(MOTOR_A_EN_PIN, speed + MOTOR_A_OFFSET);
}

void setSpeedMotorB(int speed)
{
  digitalWrite(MOTOR_B_IN_1_PIN, LOW);
  digitalWrite(MOTOR_B_IN_2_PIN, HIGH);
```

```
  // set speed to 200 out of possible range 0~255
  analogWrite(MOTOR_B_EN_PIN, speed + MOTOR_B_OFFSET);
}

void setSpeed(int speed)
{
  // this function will run the motors in both directions at a fixed speed
  // turn on motor A
  setSpeedMotorA(speed);

  // turn on motor B
  setSpeedMotorB(speed);
}

void stop()
{
  // now turn off motors
  digitalWrite(MOTOR_A_IN_1_PIN, LOW);
  digitalWrite(MOTOR_A_IN_2_PIN, LOW);
  digitalWrite(MOTOR_B_IN_1_PIN, LOW);
  digitalWrite(MOTOR_B_IN_2_PIN, LOW);
}
```

```
void drive(int distance, float turn)
{
  int fullSpeed = 230;
  int stopSpeed = 0;

  // drive to location
  int s = fullSpeed;
  if ( distance < 8 )
  {
    int wouldBeSpeed = s - stopSpeed;
    wouldBeSpeed *= distance / 8.0f;
    s = stopSpeed + wouldBeSpeed;
  }

  int autoThrottle = constrain(s, stopSpeed, fullSpeed);
  autoThrottle = 230;

  float t = turn;
  while (t < -180) t += 360;
  while (t >  180) t -= 360;

  Serial.print("turn: ");
  Serial.println(t);
  Serial.print("original: ");
```

```
Serial.println(turn);

float t_modifier = (180.0 - abs(t)) / 180.0;
float autoSteerA = 1;
float autoSteerB = 1;

if (t < 0)
{
  autoSteerB = t_modifier;
}
else if (t > 0)
{
  autoSteerA = t_modifier;
}

Serial.print("steerA: ");
Serial.println(autoSteerA);
Serial.print("steerB: ");
Serial.println(autoSteerB);

int speedA = (int) (((float) autoThrottle) * autoSteerA);
int speedB = (int) (((float) autoThrottle) * autoSteerB);

setSpeedMotorA(speedA);
```

```
  setSpeedMotorB(speedB);
}

void driveTo(struct GeoLoc &loc, int timeout)
{
  nss.listen();
  GeoLoc wagonLoc = checkGPS();
  bluetoothSerial.listen();
  GeoLoc phoneLoc = checkPhone();

  if (wagonLoc.Lat != 0 && wagonLoc.Lng != 0 && enabled)
  {
    float d = 0;
    //Start move loop here
    do
    {
      nss.listen();
      wagonLoc = checkGPS();
      bluetoothSerial.listen();
      GeoLoc phoneLoc = checkPhone();

      d = geoDistance(wagonLoc, loc);
      float t = geoBearing(wagonLoc, loc) - geoHeading();
```

```
      Serial.print("Distance: ");
      Serial.println(geoDistance(wagonLoc, loc));

      Serial.print("Bearing: ");
      Serial.println(geoBearing(wagonLoc, loc));

      Serial.print("heading: ");
      Serial.println(geoHeading());

      drive(d, t);
      timeout -= 1;
    } while (d > 3.0 && enabled && timeout>0);

    stop();
  }
}

void setup()
{
  // Compass
  compass.init();

  // Motor pins
  pinMode(MOTOR_A_EN_PIN, OUTPUT);
```

```
  pinMode(MOTOR_B_EN_PIN, OUTPUT);
  pinMode(MOTOR_A_IN_1_PIN, OUTPUT);
  pinMode(MOTOR_A_IN_2_PIN, OUTPUT);
  pinMode(MOTOR_B_IN_1_PIN, OUTPUT);
  pinMode(MOTOR_B_IN_2_PIN, OUTPUT);

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);

  //Debugging via serial
  Serial.begin(9600);

  //GPS
  nss.begin(9600);

  //Bluetooth
  bluetoothSerial.begin(9600);

  Wire.begin();
}

// Testing
void testDriveNorth() {
  float heading = geoHeading();
```

```
  int testDist = 10;
  Serial.println(heading);

  while(!(heading < 5 && heading > -5)) {
    drive(testDist, heading);
    heading = geoHeading();
    Serial.println(heading);
    delay(500);
  }

  stop();
}

void loop()
{
  GeoLoc wagonLoc = checkGPS();
  nss.listen();
  GeoLoc phoneLoc = checkPhone();
  bluetoothSerial.listen();
  driveTo(phoneLoc, GPS_STREAM_TIMEOUT);
}
```

The code for the definitions for our project is as follows:

```
#define GPS_TX_PIN 6

#define BLUETOOTH_TX_PIN 10
#define BLUETOOTH_RX_PIN 11

#define MOTOR_A_EN_PIN 5
#define MOTOR_B_EN_PIN 9
#define MOTOR_A_IN_1_PIN 7
#define MOTOR_A_IN_2_PIN 8
#define MOTOR_B_IN_1_PIN 12
#define MOTOR_B_IN_2_PIN 4

// If one motor tends to spin faster than the other, add offset
#define MOTOR_A_OFFSET 20
#define MOTOR_B_OFFSET 0

// You must then add your 'Declination Angle' to the compass, which is the 'Error' of the magnetic field in your location.
// Find yours here: http://www.magnetic-declination.com/
// Mine is: 13° 24' E (Positive), which is ~13 Degrees, or (which we need) 0.23 radians
#define DECLINATION_ANGLE -0.12f
```

```
// The offset of the mounting position to true north
// It would be best to run the /examples/magsensor sketch and compare to the compass on your
 smartphone
#define COMPASS_OFFSET 0.0f

// How often the GPS should update in MS
// Keep this above 1000
#define GPS_UPDATE_INTERVAL 1000

// Number of changes in movement to timeout for GPS streaming
// Keeps the cooler from driving away if there is a problem
#define GPS_STREAM_TIMEOUT 18

// Number of changes in movement to timeout for GPS waypoints
// Keeps the cooler from driving away if there is a problem
#define GPS_WAYPOINT_TIMEOUT 45

// Definitions (don't edit these)
struct GeoLoc {
  float lat;
  float lng;
}
```

Figure 14: “Follow me” code and code definitions.

DISCUSSION AND CONCLUSION:

The very first result we had was the creation of our Android app using the MIT App Inventor. Although the app is tailored for Android, we procured an Android device for testing purposes at an affordable cost. By integrating the HC-05 Bluetooth module, our app could efficiently transmit real-time GPS location data from the user's Android smartphone to the Arduino. We encountered a minor initial delay in data transmission during app implementation, which was promptly resolved, ensuring a smoother and improved communication channel between the phone and the Arduino. The second pivotal aspect of our project focused on enabling the Wheelchair to precisely determine its location and heading. We meticulously addressed and implemented this goal. Our coding for the GPS and compass components equipped the Wheelchair with the capability to calculate its location and direction effectively. This accomplishment played a crucial role in our project, establishing the foundation for seamless operation and effective coordination between the Wheelchair and the user's smartphone app.

In conclusion, the primary objective of this project was to reduce the necessity for manual effort to the fullest extent, and it is with great satisfaction that we can affirm this goal has been successfully accomplished. The integration of cutting-edge technology, such as the app generated using the MIT App Inventor, has significantly enhanced the efficiency and ease of operating the electric wheelchair. The app functions seamlessly, continuously transmitting GPS coordinates to the Arduino, ensuring precise and responsive control. It is important to acknowledge that while this system has proven to be highly functional, there is still ample room for further improvement. Future work may involve the incorporation of additional sensors or control mechanisms, such as a joystick, to enhance user experience and expand the capabilities of the electric wheelchair.

FUTURE WORK:

Even thought the project mainly focused on the driving and communication part of the wheelchair, a future work would be the addition of sensors for a safe and comfortable drive. Ultrasonic sensors are highly valuable for obstacle avoidance in an automated wheelchair. They are reliable, cost-effective, and capable of detecting obstacles at various distances. By strategically placing ultrasonic sensors around the wheelchair, you can ensure efficient obstacle detection and navigation, enhancing user safety and convenience. Additionally, Gyroscope sensors are crucial for maintaining balance and stability in the automated wheelchair. They help prevent tipping and ensure smooth movements, especially when navigating uneven terrain. In addition to the above considerations, it's essential to address potential issues related to GPS accuracy, particularly when the electric wheelchair is used indoors or in areas with poor GPS signal reception. GPS signals can be obstructed by buildings or other structures, leading to inaccuracies in the wheelchair's position tracking. To mitigate this challenge, it would be advantageous to incorporate an alternative control method, such as a joystick, into the overall wheelchair system. A joystick control offers several benefits, including enhanced maneuverability and real-time control regardless of GPS signal strength. It allows the user to have more direct and precise control over the wheelchair's movements, making it a valuable addition in scenarios where GPS data may not be readily available.

REFERENCES:

[1] Groover, M. P. (2019). Automation. In *Encyclopædia Britannica*. https://www.britannica.com/technology/automation

[2] IBM. (2021). What is automation? Www.ibm.com. https://www.ibm.com/topics/automation

[3] Assistive technology. (n.d.). Www.who.int. https://www.who.int/health-topics/assistive-technology#tab=tab_1

[4] Design of an arduino based voice-controlled automated wheelchair. IEEE Conference Publication| IEEEXplore. Published December 1, 2017. https://ieeexplore.ieee.org/document/8288954

[5] "All about HC-05 Bluetooth Module | Connection with Android," *GeeksforGeeks*, Jun. 03, 2020. https://www.geeksforgeeks.org/all-about-hc-05-bluetooth-module-connection-with-android/

[6] L. S. Y. Dehigaspege, M. K. C. Liyanage, N. A. M. Liyanage, M. I. Marzook, & Dhammearatchi, D. (2017, July 08). Follow me Multifunctional Automated Trolley. Retrieved February 1, 2023, from https://www.ijert.org/follow-me-multifunctional-automated-trolley

[7] "L298N Motor Driver Module," *Components101*. https://components101.com/modules/l293n-motor-driver-module

[8] Shack, H. (2022, December 26). Make an Autonomous "Follow me" Cooler. Retrieved February 1, 2023, from https://www.hackster.io/hackershack/make-an-autonomous-follow-me-cooler-7ca8bc#code

[9] Assistive technology. (n.d.). Www.who.int. https://www.who.int/health-topics/assistive-technology#tab=tab_1