Scoot Free

University Project - 2
Submitted by
T104

Abbas Ali Pathan	**20171CSE0005**
Sadiya Kauser	**20171CSE0595**
Sameera Noorian	**20171CSE0607**
Saquib Ameen Ulla Shariff	**20171CSE0620**
Syed Shahbaz	**20171CSE0708**

Under the guidance of

Ms. Nikita

in partial fulfillment for the award of the degree

of

Bachelor of Technology

IN

Computer Science and Engineering

At

Department of
Computer Science and Engineering

School of Engineering

PRESIDENCY UNIVERSITY

BANGALORE

MAY 2021

CERTIFICATE

This is to certified that the Project report "**Scoot Free**" being submitted by Abbas Ali Pathan : 20171CSE0005, Sadiya Kauser: 20171CSE0595, Sameera Noorian: 20171CSE0607, Saquib Ameen Ulla Shariff: 20171CSE0620, Syed Shahbaz: 20171CSE0708, in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Mohan K G
HOD
Department of CSE
Presidency University

Ms. Nikitha
Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING

PRESIDENCY UNIVERSITY

DECLARATION

I hereby declare that the work, which is being presented in the project report entitled **Scoot Free** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Ms. Nikitha,** Assistant Professor**, Department of Computer Science and Engineering**, School of **Engineering**, Presidency University, Bangalore.

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Abbas Ali Pathan _____
20171CSE0005

Sadiya Kauser _____
20171CSE0595

Sameera Noorian _____
20171CSE0607

Saquib Ameen Ulla Shariff _____
20171CSE0620

Syed Shahbaz _____
20171CSE0708

ABSTRACT

Automation is the use of machines, control systems and information technologies to optimize productivity in the production of goods and delivery of services. An automated porter is an Arduino Uno powered robot which follows an individual with his/her belongings placed on it. It connects to a smart android device via Bluetooth and navigates by comparing its own location to the user's android device location using GPS. It carts objects of allowed weight and dimensions that follows an individual enabling a handsfree walk. This lets one concentrate and perform other vital work functions while on the move. This autonomous porter facilitates Senior citizens and specially disabled individuals to circumvent the challenge of carrying their belongings where one can avoid hurting or spraining themselves by securely carrying their possessions, hence it makes the task effortless and efficient. In terms of assurance, it uses a biometric fingerprint scanner which secures one's belongings and uses an alarm system to alert the user when disconnected. In order to overcome the above said problems, the smart porter has been designed.

List of Tables

List of Figures

TABLE OF CONTENTS

ACKNOWLEDGEMENT

First of all, We indebted to the GOD ALMIGHTY for giving me an opportunity to excel in our efforts to complete this project on time.

We are extremely grateful to **Dr. C. Prabhakar Reddy**, Dean, School of Engineering and **Dr.Mohan K G**, Head of the Department, Department of Computer Science and Engineering, for providing all the required resources for the successful completion of this project.

We sincerely thank our project guide, **Prof. Nikita**, for his guidance, help and motivation. Apart from the area of work, we learned a lot from him, which we are sure will be useful in different stages of our life. We would like to express our gratitude to Faculty Coordinators and Faculty, for their review and many helpful comments.

We would like to acknowledge the support and encouragement of our friends.

Abbas Ali Pathan

Sadiya Kouser

Sameera Noorian

Saquib Ameen Ulla Shariff

Syed Shahbaz

Chapter 1 : Introduction

1.1 Introduction to IOT

The Internet of Things is an emerging topic of technical, social, and economic significance. Consumer products, durable goods, cars and trucks, industrial and utility components, sensors, and other everyday objects are being combined with Internet connectivity and powerful data analytic capabilities that promise to transform the way we work, live, and play.

How are you reading this report right now? It might be on desktop, on mobile, maybe a tablet, but whatever device you're using, it's most likely connected to the internet.

An internet connection is a wonderful thing, it give us all sorts of benefits that just weren't possible before. If you're old enough, think of your cell phone before it was a smartphone. You could call and you could text, sure, but now you can read any book, watch any movie, or listen to any song all in the palm of your hand.

The point is that connecting things to the internet yields many amazing benefits. We've all seen these benefits with our smartphones, laptops, and tablets, but this is true for everything else too. And yes, we do mean everything.

The Internet of Things is actually a pretty simple concept, it means taking all the physical places and things in the world and connecting them to the internet. Internet of Things devices are definitely going to play a really important role in future technology advancements. Although there are still the same issues that have to be addressed. In fact, one of the main concerns about IOT devices can be cybersquatting.

Because most IOT devices make use of a cloud centre to store their data and to collect useful information from the internet, that makes them vulnerable from Hackers attacks (creating a single point of failure).

In order to resolve this problem, could be either possible to increase the encryption standards (slowing down the transfer of data) or makes use of Artificial Intelligence security powered techniques such as Differential Privacy and Federated Learning.

There are four main components used in IOT:

1.Low-power embedded systems –

Less battery consumption, high performance are the inverse factors play a significant role during the design of electronic systems.

2.Cloud computing –

Data collected through IOT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.

3.Availability of big data –

We know that IOT relies heavily on sensors, especially real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.

4.Networking connection –

In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

There are two ways of building IOT:

1.Form a separate internet work including only physical objects.

2.Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

1.2 Introduction to Scoot Free

In today's era, human lifestyle has nourished to be more complex and tangled with restlessness. One would want to retain each and every minute in a way where one cannot be intervened by tasks which are less prioritized compared to others . The main motto of Scoot Free is to offer comfort and ease while engaged with medical treatments, travelling, shopping and so on.

The smart porter ports all your belongings and one can opt an additional mod for security of their belongings. The smart porter doesn't jeopardize an individual's space and privacy as in case of a Coolie. One can also deliver goods within the limited range by assigning the desired GPS way points. This Arduino Uno powered machine is designed and built to be cost effective for an invention that can innovate various applications to aid humans. This autonomous porter facilitates Senior citizens and specially disabled individuals to circumvent the challenge of carrying their belongings where one can avoid hurting or spraining themselves by securely carrying their possessions, hence it makes the task effortless and efficient.

The autonomous porter can be controlled by operating an application built using Android Studio. This application is designed to have a universal usable interface. The goal of universal usability is to enable the widest possible range of users to benefit from this particular application. This goal is stronger than merely providing access, which focuses on technology availability and is often tied to access for users with disabilities. The application tends to be consistent and simple to use with minimal steps provided. The application consist of two functions that is GPS (Global Positioning System) continuous streaming and fixed GPS way points. This is an Android platformed specified application which can be used by users irrespective of the age and culture.

The smart porter links to the android device via HC-05 Bluetooth module and navigates by comparing its own location by using NEO-6M GPS module to the user's android device location using GPS. The smart porter is compatible to run only on flat surfaces.

The automated porter can be used either as a trolley or as a luggage container.

The porter can be used in huge industries like computer industries, agriculture industries, food industries, pharmaceutical industries and enterprises so as to carry their required tools and goods by an individual worker. The porter can be further used in health care centers where surgical technologist can anticipate the needs of the surgeon by adding instruments and providing needed supplies such as sponges, sharps and instruments via porter. The porter can also be used as a delivery robot, also as rescue food supplier. Looking deeply into environment or our surroundings, we will be able interpret that “YES” there is a need of such robot that can assist humans and can serve them. Such a robot can be used for many purposes. With few changes, the robot can act as a human companion as well.

The porter is also provided with a biometric fingerprint scanner in need of securing your personal belongings. The user will be notified as soon as an individual disconnects from his/her smart porter by an alarming buzzer.

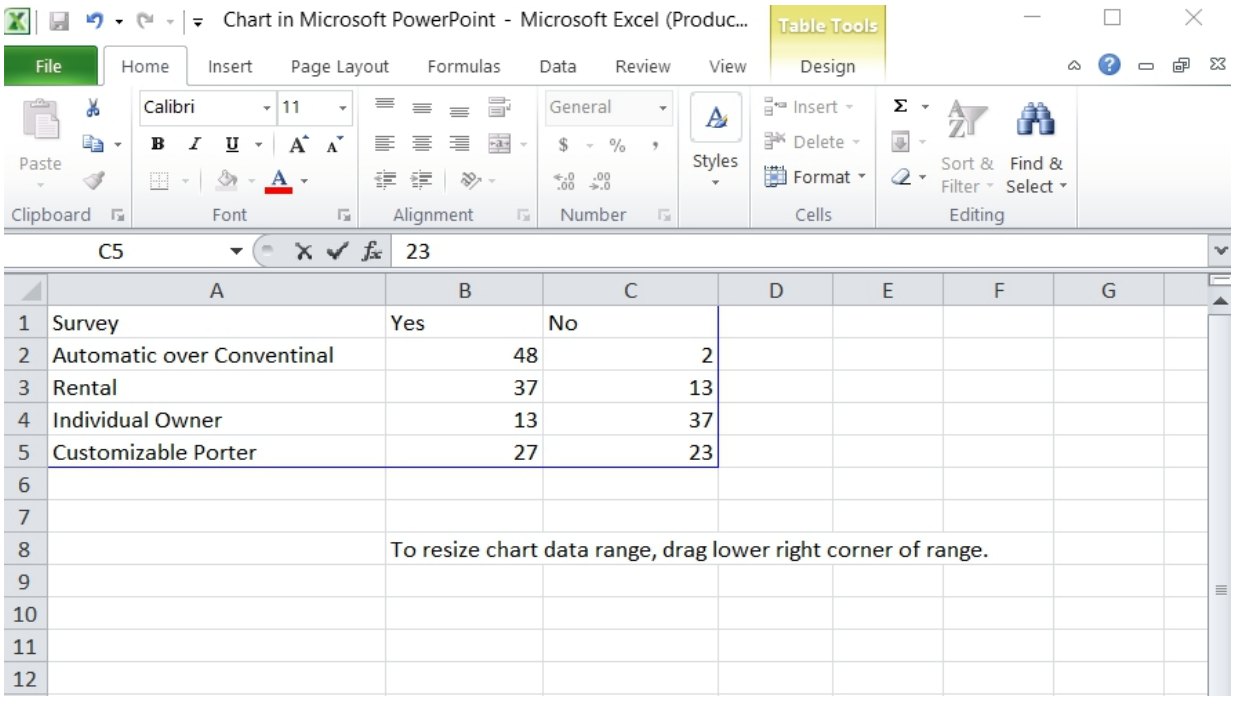
Chapter 2: Requirement analysis

Carting belongings or luggage has always been challenging, tiring and time consuming chore. Some of the common phenomenons faced while travelling or shopping are tripping over the belongings due to heavy weight , hurting or spraining by carrying the luggage , damaging of our goods or luggage, delaying of centralized tasks while engaged in carrying the belongings, misplacing of belongings , theft and insecurity . In order to determine the objective of the smart porter a survey was conducted by our team.

The following set of questions were put forward to a set of individuals :

1. Do you think it's convenient and efficient for one to use?
2. Does it make your life easier?
3. If you could change one thing about our product for refinement?
4. If Agreed, What would that be?
5. How strongly do you agree that the smart porter is better than the conventional porter ?
6. What do you think the pricing should be ?

Survey Data :



	A	B	C	D	E	F	G
1	Survey	Yes	No				
2	Automatic over Conventinal	48	2				
3	Rental	37	13				
4	Individual Owner	13	37				
5	Customizable Porter	27	23				
6							
7							
8		To resize chart data range, drag lower right corner of range.					
9							
10							
11							
12							

Fig 2.1 Survey data

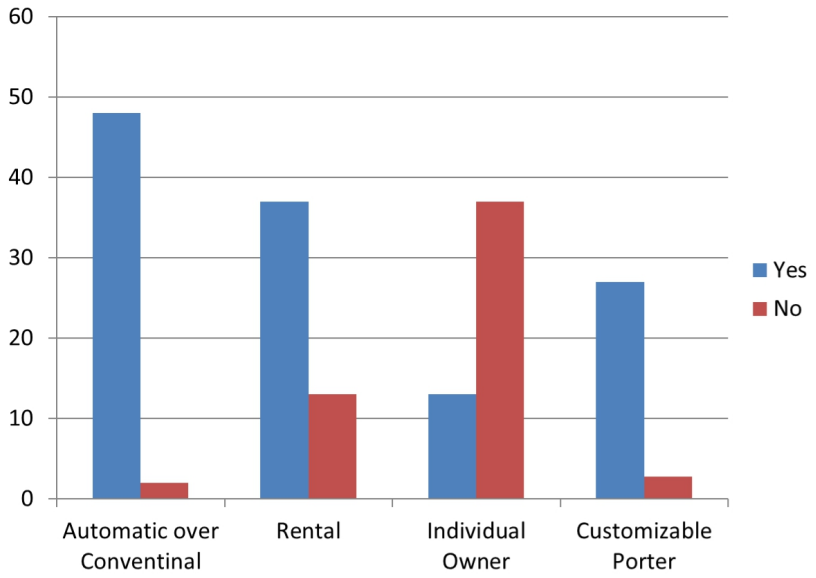


Fig 2.2 graphical representation of the survey

The Fig : 2.2 is a graphically representation of the survey conducted where 96% of the individuals preferred an automated porter over the conventional porter because one shouldn't have to compromise their personal space or feel uneasy for an additional stranger(Coolie), it is an efficient and easy going Arduino powered machine, accommodates more goods at a go, an individual's personal belongings are more secure, one can prevent from harming themselves and also saves plenty of time, users with toddlers or infants often find it challenging to manage their luggage or belongings. 74% of the individuals opted for renting the porter than individually owning it because they believed, as it is more compatible on the flat surfaces as it would be more convenient to lend the porter in malls, marts, airports and so on, as these places have flat flooring, usually the average space in houses isn't enormous in order to use the porter, since, it isn't frequently used in household works. 54% of the individuals desired for a customization porter because an individual preferred it to be compact but also flexible in a way where one can expand or contract the porter according to the need faced, the smart porter acquires minimal space.

Chapter 3 : Literature review

3.1 Reports of Existing Project 1 (Sonali Patil, Shrutika Patil, Anuja Patil, Prof. Deshmukh S. C)

In this paper, we explained how robots can act in concert with human behavior. Our aim is to develop a robot capable of maneuvering through busy airports behind its owner while hauling his or her luggage. In this paper, In order to follow a human, a mobile robot needs to know the position of the person and must be able to determine its own path in order to follow his target. We consider a method using an transmitter &receiver. In order to prevent collision with obstacles, ultrasonic sensors are used to detect objects that may be in its path. We present the effectiveness of our approaches by showing the experimental result using a real platform. In this article we describe the research carried out in the attempt to develop a human-following robot. Automatic or automation means, as by electronic devices, reducing human intervention to a minimum. This will reduce the time delay and human efforts in luggage management system.

Now a days-everybody uses a luggage for travel especially to airport all of them dragging out heavy luggage. Passenger need to carry his /her own luggage. This is very slow and expensive process. And it becomes hectic journey. This problem can be overcome by automatic luggage follower system. It is nothing but smart luggage. It reduces the time delay and human efforts in luggage management. For the implementation of design ultrasonic sensor and dc motors plays important role. For the anti-theft tracking purpose GSM and GPS are used. Whenever the bag is lost, the user can access the location by GSM and GPS tracking system .Another feature in this system is dry batteries. Generally lithium batteries are used for battery pack. But lithium batteries catch fires, when it punctured. So it is harmful for system. And dry batteries are rechargeable and easy to carry. Objectives are:A luggage easy to be carry and to be manageable by any person, A more way to carry the luggage in case of any problem,Comfortable cost according to everyone’s perspective, A luggage with an attractive and innovative exterior design, A security system that the user can be free of worries of his or her luggage.

Where ever the people travel they used to carry luggage especially through airports all of them dragging out their heavy luggage. Perhaps trailing of the bag is very difficult task for old peoples. If a bag that follows passengers by utilizing human following concept, then entire problem gets vanished following technique is implemented using data taken from two ultrasonic sensors. Ultrasound sensors always measure distance between bag and human by sending sound waves and collects the reflected waves when it strikes an obstacle. The forward, right and left movement of the bag is based on the signal strength received at each receiver section.

Target Detection:Detection is the process of finding both position user and robot. Here we are using ultrasonic sensor. The ultrasonic sensor is the key element in target detection. Ultrasound sensor measure distance by using ultrasound waves. It is based on the properties of acoustic waves with frequency above human audible range, PIC:Peripheral interface controller with 40 DIP is used in automatic luggage follower for controlling operations. It operates in 5.5V supply voltage with internal oscillator frequency 16MHz. PIC operates with 75 instructions and having USART pin for serial communication.Obstacle Detection: IR sensor. The obstacle detection is done on the robot by providing an infra red eye to it. The infra red eye is the IR transceiver circuit through which the robot detects the obstacles and controls itself either by changing direction for stationary obstacles or by stopping itself.

GPS: The Global Positioning system is one of the systems that are space based radio navigation system made up of at least 24 satellites. It is nothing but a global navigation satellite system that provides Geo-location and the time information to a GPS receiver anywhere on or near the earth where there is an unobstructed line of sight to four or more GPS satellite.

GSM module: GSM is a standard developed to describe second generation digital cellular protocols used by mobile phones. It is a TDMA (Time division multiple access) based network technology.

DC Motor: A DC motor is any of a class of rotating electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. The speed DC motor can be controlled by varying the supply voltage or by changing the current in its field winding.

In day to day life when we are traveling luggage carrier is big problem. Using this technique we can overcome this problem. A security system that the user can be free of worries of his or her luggage being stolen or left behind. In future we will add features like headphone points, USB point, Wi-Fi technology, fingerprint system for security purpose. And focus on to make less expensive and easy to handle.

3.2 Reports of Existing Project 2 (Vikas, Vikash Kumar, Vikram Kumar, Umesh Nath Dr. Prateek Singhal Associate Professor)

This project produced an Automatic Trolley Human Follower for general or industrial user. An automatic trolley human follower is developed to help a user or production industry. This automatic trolley human follower is controlled by a PIC 16F877A micro controller that can follow the user automatically with integrated circuit of ultrasonic and IR sensor. The 12V DC motor was used as the power supplied to move the trolley automatically follow a user in hypermarket or industrial use. In this project a robotic vehicle is fabricated which runs like a regular trolley by carrying tools from place to another to reduce the utilization of human energy in order to carry heavy things. This is done by using a set of receiver and transmitter ultrasonic sensor to detect a user in a 1 meter range and it will follow the user.

Shopping trolleys are available in the shopping mall which are wheeled and are to be carried by the person. The shopping trolleys are available in various sizes and with baby sitters.

Trolleys are fitted with the castor wheels and normal wheels for easy to move on the floor while shopping. Some people are uncomfortable to carry the trolley since it is tedious and uncomfortable to push or pull it in the crowd. We are proposing to make the automatic trolley for shopping mall which can sense us and follow us. The beauty of all this system is that it uses all of these sensors in the most effective way to help it react. Unmanned operation requires sensor system for target position, sensors for load position, and control and communication equipment on the trolley and remote consoles for control signals. To remain fully operational with the terminal at all times

requires extraordinary work flows, narrow time schedules and lots of work which do considerably drive up the overall investment. Now a day, automatic trolley has become popular especially in localization scheme. It is a non-touching recognition system where it can tag and send tag data wireless at various distances. In order to prevent objects collision, ultrasonic sensor, light dependent resistor was used in this project.

The project aims to develop an automatic human guided shopping trolley with a smart shopping system. This shopping trolley can lead a user to the items' locations in supermarket and he or she is able to know the items' locations through a shopping map. It will follow user.

3.3 Reports of Existing Project 3 (L.S.Y. Dehigaspege, M.K.C. Liyanage, N.A.M. Liyanage, M.I. Marzook, and Dhishan Dhammearatchi)

Shopping carts in the supermarkets in day today shopping activities is now mostly visible. Customers are pushing trolleys around them to carry the items they purchased. The usual process of travelling the trolley is done manually by the human with the effort of his/her. Therefore, if a customer carries a baby while doing shopping it is a real burden to the customer to push the trolley or to a disabled person with one hand is almost impossible to push the trolley. People can see huge rush in supermarkets on holidays and weekends the rush is even more when there are special offers and discounts. The main purpose of the research project was to address the above issues by developing a multi-functional automated trolley. Follow Me is an automated trolley that is capable of carrying goods while following the customer automatically without human effort. Supermarkets in the world plays a major role when it comes for shopping and the supermarkets which exist in the market implement many things to compete with other supermarkets. "Follow Me" automated smart trolley provides some benefits to the supermarkets as well as to the customers. "Follow Me" trolley consist of some multifunctional tasks such as it is able to follow the customers automatically so that the customer does not needs to push the trolley manually. "Follow Me" has an android application which helps the customers to track the goods details tasks which the android

application perform has been described in section III part D. An important event that this trolley performs is the automatic parking. After the customer finished their purchases the trolley move back to its slot automatically without any helpmate. Line following methodology was used by the research group to implement the automatic parking technology. All the above mentioned functions and technologies were used to fulfil all the objectives of this "Follow Me" Multifunctional Automated Trolley and the research group hope the research would be a benefit for the developing supermarkets. Hope that this research study will be helpful for the researchers who interested in the automated systems as well as software building and will develop similar models with more advance technologies and features.

3.4 Reports of Existing Project 4 (Afrin Khan, Bandini Nalwade, Neha Kharshinge, Sonali Kamble 1,2,3,4 Student, Dept. of Computer Science and Engineering, Dr. J J Magdum College of Engineering, Maharashtra, India)

This project targets all sections of society and all age groups. Whether they are of old age or young, carrying a heavy luggage has always been a matter of distress. This project aims to provide comfort and ease while travelling in addition with smart features that are user friendly. The carrier is designed in such a way that it follows a specific user who is having RF transceiver and in addition with a feature of avoiding obstacles in the path by raising an alarm on the user's smart phone. There is a facility of tracking the location of the luggage too so as to avoid theft and loosing the luggage. The location of luggage will send on user's smart phone via a message. The problems encountered have been divided into following three sections: Back pain Problem, Accidental injury if backpack falls, Shoulder and neck injury, Weakening of cuff bones and joins.

Luggage is a place to keep things of its user in transit. It has many design changes since its origin. All the changes have made the luggage easily transportable. Each time the luggage had an upgrade it transformed its handling facilities but it also made itself vulnerable to theft, and also causing many problems to its user. In present day, the luggage is not compatible with

the current smart life of the people. In Airports, a lot of luggage do out clearance because of lack of knowledge and ignorance of baggage rules. In order to overcome the above said problems the luggage is been designed from inside out.

Smart bag is an innovative carry on suitcase that makes life easier and smoother. Carrying luggage is the main difficulty faced by each and every passenger. Here we try to solve the dragging of luggage difficulty and also providing better security and intelligent features that suitable for modern era. In this project we developed a new low cost human following technology to assist low cost consumer product implementation, so that the overall production cost of a automatic user following bag will be less. The inbuilt power bank can provide sufficient power and at the same time share power to users gadgets like smart phone, laptops etc.

Chapter 4: Existing System

[1] There are plenty of machines which aren't generalized rather these machines are focused on one particular function .

The Dash Cart was introduced by Amazon in the Woodland Hills neighborhood of Los Angeles. It has smart Amazon-made grocery carts for an individual to use and to input the price look-up code of the item before you place it in the cart to be weighed and added to your order.

The cart uses a combination of computer vision algorithms and sensor fusion to identify items you put in the cart. When you exit through the store's Amazon Dash Cart lane, sensors automatically identify the cart, and your payment is processed using the credit card on your Amazon account.

There are also prototype cart models(not real time model) which can be controlled via hand gestures using infrared sensors.

[2] The conventional luggage bag has its own importance and drawbacks and this lead to the invention of luggage which tracks its location, which follows the user manually but can be found when lost. It highlights the GPS (Global Positioning System) tracker, built-in emergency power bank. While providing convenience to the goods carriers, the prototype also features where the GPS device is used to track the luggage bag which has a built-in power bank that makes the luggage bag a charging station for gadgets.

[3] Micro, a Swiss company known for cutting-edge scooters and kick boards, has made the leap into travel gear, combining a carry-on-sized suitcase with a three-wheeled scooter. The interior has all sorts of neat pockets, but the functionality falls apart when it comes to packing efficiently. You must pay special attention to how you load it, placing heavy stuff in the back and light stuff in the front, to prevent it from tipping over when you're riding it. Also, the handle is large, so you can grip it like a scooter's handles as you ride it, but this means you can't slide another bag — such as a laptop bag — over the handle to pull them both at once. It's not suitable for heavy travel use. It's too small and doesn't accommodate enough weight to be useful for someone who needs to pack, say, a large laptop, a tablet, a camera and other carry-on essentials. Machines with other features such as line following robot, motion detection robot, remote controlled suitcase but all had their own limitations.

Chapter 5: Proposed System

The Scoot Free automated porter is designed after analyzing the past modes of porter. Carrying or securing one's belongings while travelling or shopping has always been a vital necessity for humans, this lead to the invention of carriers, bags, suitcases which was further upgraded to bags and suitcases with locking system which further got ameliorate with better security system that is inbuilt 3-digit locker. Generally, moving or carrying commodities of more than 2 Kilograms has always been a straining and stressful task and drains a lot of energy even if it is carried by a trolley or contained in a baggage, one still needs an external human force in order to accomplish the following work. Among all the modes of carrying system, suitcases and trolleys have been the most efficient one's but not completely eases an individual, it still leads to intervening by hurting or slowing down an individual.

Keeping this in mind the smart porter is designed. The smart porter ports all your belongings and one can opt an additional mod for security of their belongings. Arduino powered robot links to the android device via HC-05 Bluetooth module and navigates by comparing its own location by using NEO-6M GPS module to the user's android device location using GPS. The smart porter has two modes, GPS streaming and GPS waypoint modes.In the GPS streaming mode, the porter will follow an individual by streaming coordinates from one's smart device. In the GPS waypoint mode, one can predetermine longitudinal and latitudinal coordinates as the destination points to the porter. Hence, the automated smart porter delivers goods within the limited range as it uses a GY-273 HMC5883L compass module and NEO-6M GPS module. The autonomous porter is powered by an Arduino Uno ATmega328p micro-controller where the source code is dumped using an Arduino IDE compiler.

The autonomous porter can be controlled by operating an application built using Java and XML on Android Studio. This application is designed to have a universal usable interface. The goal of universal usability is to enable the widest possible range of users to benefit from this particular application. This goal is stronger than merely providing access, which focuses on technology

availability and is often tied to access for users with disabilities. The application tends to be consistent and simple to use with minimal steps provided.

The application consist of two functions that is GPS(Global Positioning System) continuous streaming and fixed GPS way points. This is an Android platformed specified application which can be used by users irrespective of the age and culture. The porter is also provided with a biometric fingerprint scanner in need of securing your personal possessions. The user will be notified as soon as an individual disconnects from his/her smart porter by an alarming buzzer.

A simple algorithm has been explained using the given figure(Fig 5.1) below :

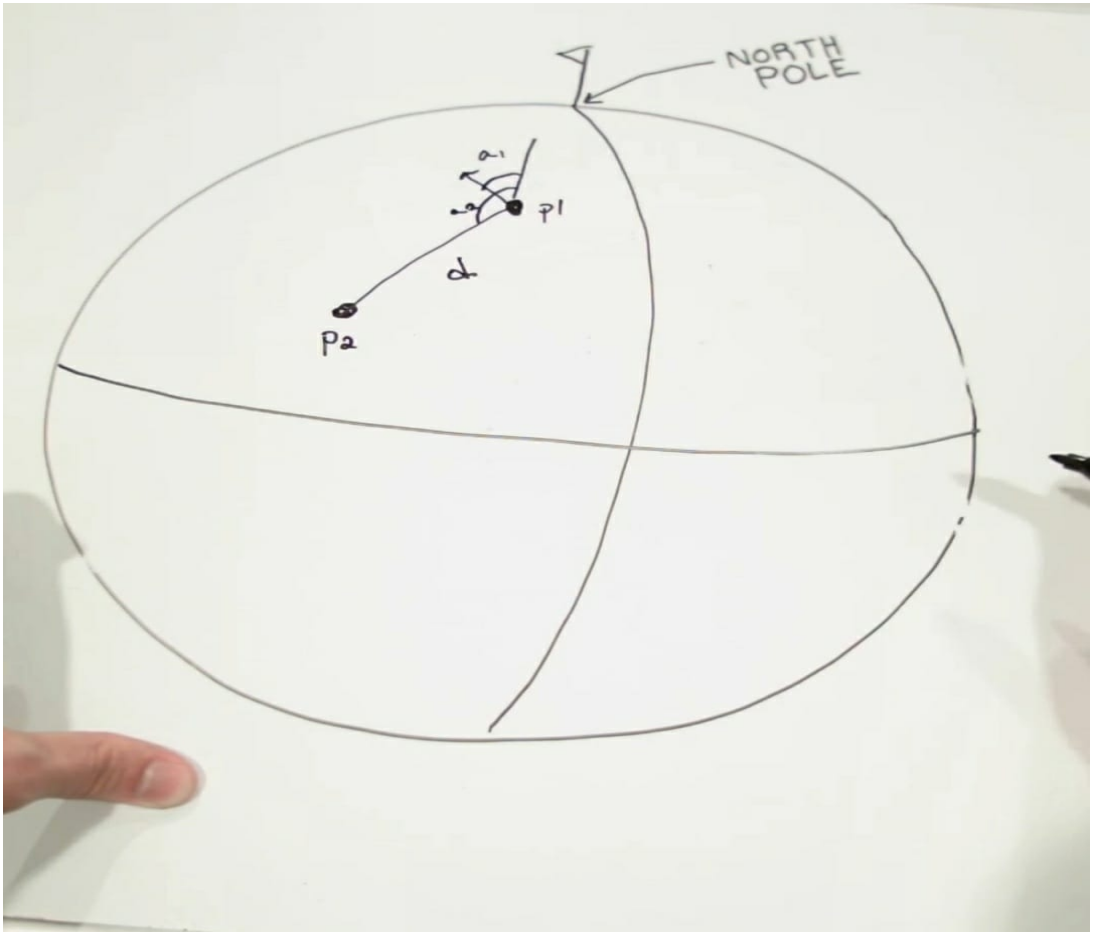


Fig 5.1 Algorithm

For an instance, consider the sphere as the earth and the top of the sphere as the North Pole.

Consider the point P1 as the porter's position, using the HMC5883l compass we can determine the direction we are pointing to. Also we can know the angle with respect to the North Pole(a1). Consider P2 as the position of an Android device with respect to the given points, we can easily find the distance between P1 and P2 using the common distance formula.

Chapter 6: System Design

The general concept of automated smart porter is to follow the user and port all your belongings leading to the decrement of efforts and increment of ease in life.

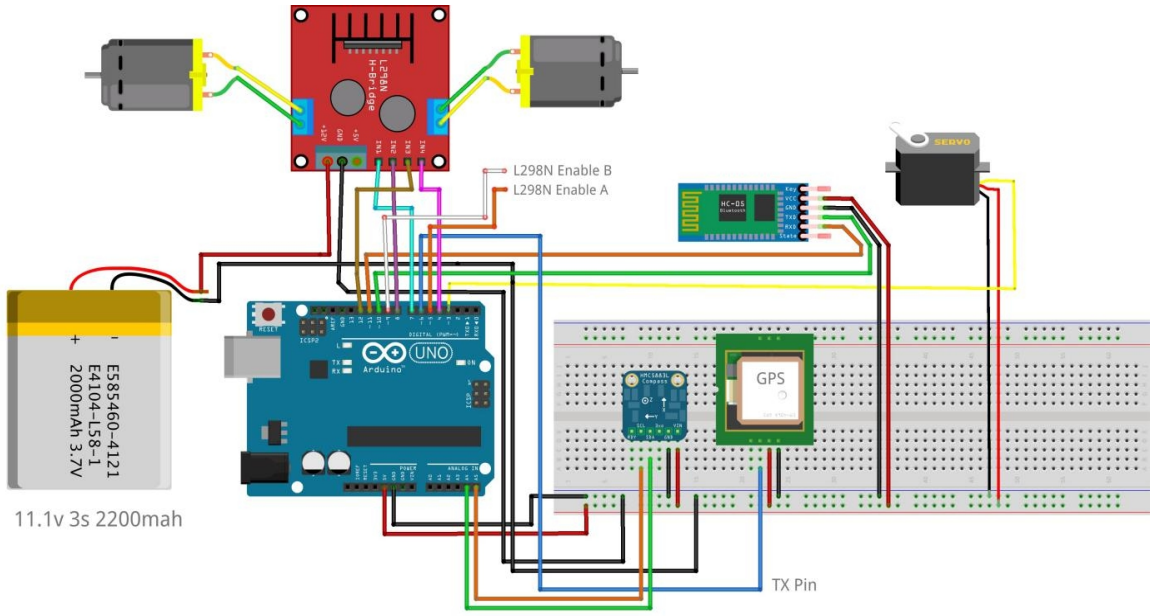


Fig 6.1 Circuit Diagram

Arduino Uno:

Pin Description:

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.

Reset	Reset	Resets the micro-controller
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Arduino Uno Technical Specification :

Micro-controller	ATmega328P – 8 bit AVR family micro-controller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Other Arduino Boards

Arduino Nano, Arduino Pro Mini, Arduino Mega, Arduino Due, Arduino Leonardo

Overview

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

How to use Arduino Board

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in Arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- Serial Pins 0 (Rx) and 1 (Tx): Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- External Interrupt Pins 2 and 3: These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM Pins 3, 5, 6, 9 and 11: These pins provide an 8-bit PWM output by using analogWrite() function.
- SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.
- In-built LED Pin 13: This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with analog Reference() function.

- Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.

Arduino Uno has a couple of other pins as explained below:

- AREF: Used to provide reference voltage for analog inputs with analogReference() function.
- Reset Pin: Making this pin LOW, resets the microcontroller.

Communication

Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The ATmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The ATmega16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. There are two RX and TX LEDs on the Arduino board which will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328P also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

Applications

- Prototyping of Electronics Products and Systems
- Multiple DIY Projects.
- Easy to use for beginner level DIYers and makers.

- Projects requiring Multiple I/O interfaces and communications.

HMC5883l (3-axis compass):

The Compass Module 3-Axis HMC5883L is a low-field magnetic sensing device with a digital interface. The compass module converts any magnetic field to a differential voltage output on 3 axes. This voltage shift is the raw digital output value, which can then be used to calculate headings or sense magnetic fields coming from different directions. The module is designed for use with a large variety of microcontrollers with different voltage requirements.

Features

- 3-Axis magneto-resistive sensor
- 1 to 2 degree compass heading accuracy
- Wide magnetic field range (+/-8 gauss)
- Fast 160 Hz maximum output rate
- Precision in-axis sensitivity and linearity
- Measures Earth's magnetic field, from milli-gauss to 8 gauss.

Key Specifications

- Power Requirements: 2.7 to 6.5 VDC
- Communication Interface: I²C (up to 400 kHz)
- Operating temperature: -22 to +185 °F
  (-30 to +85 °C)
- Dimensions: 0.73 x .65 in (1.8 x 1.7 cm)

Application Ideas

- Auto and personal navigation
- UAV systems
- Robotic navigation
- Location-based services (LBS)

Resources and Downloads

Check for the latest version of this document, free software, and example programs from the Compass Module 3-Axis HMC5883L product page. Go to www.parallax.com and search for part number 29133.

Quick-Start Guide

The following is a very basic procedure to get started initializing and reading values from the Compass Module. This module is equipped with 4.7 kΩ pull-up resistors on the SDA line and a 2.2K resistor on the SCL line. There is no need for any external hardware to operate. The schematic is available from the 29133 product page at www.parallax.com.

1. With main power off, make the proper connections between the compass module and a

microcontroller as shown above. The voltage connected to the VIN pin should be the same as the voltage powering the microcontroller communicating with the device.

2. Power on the device and load the BASIC Stamp or Propeller microcontroller sample code

provided on the 29133 product page.

Note: For other microcontrollers first, setup the module for continues output mode by writing (0x3C) 0x00 to the MODE register (0x02). Once running in continues mode set a pointer to OUTPUT_X_MSB by writing (0x3C) 0x03 to the device. Now the compass module will output raw data from OUTPUT_X_MSB - OUTPUT_Y_LSB. The compass will automatically index through these registers; there is no need to set pointers for each of these registers. The last step would be to read from each of the six output registers, 8 bits of data per register. For more details on these modes and registers please see the datasheet on the 29133 product page at www.parallax.com.

Precautions

This module is VERY sensitive to ferrous material, metal objects, magnets and power supplies, etc. Any of these items in close range may affect performance. If you're experiencing adverse results, check your environment for anything that may influence magnetic fields. It is recommended to connect the unit with wires, keeping it away from a breadboard.

Without tilt compensation the device will need to be level for accurate headings. For tilt compensation, you will need an accelerometer. See the application notes available on the 29133 product page for details on tilt compensation.

Modes of Operation

The Compass Module has several modes that can be accessed from the I²C bus, whose primary purpose is power management. All the modes are controlled by the Mode Register. The following section describing modes, was taken from the Honeywell HMC5883L data sheet.

.

Continuous-Measurement Mode

During continuous-measurement mode, the device continuously makes measurements, at user select-able rate, and places measured data in data output registers. All registers maintain values while in continuousmeasurement mode and I²C bus is enabled for use by other devices on the network.

Single-Measurement Mode

This is the default power-up mode. During single-measurement mode, the device makes a single measurement and places the measured data in data output registers. All registers maintain values while in single-measurement mode. The I²C bus is enabled for use by other devices on the network while in single-measurement mode.

Idle Mode

During this mode the device is accessible through the I²C bus, but major sources of power consumption are disabled, such as, but not limited to, the ADC, the amplifier, and the sensor bias current. All registers maintain values while in idle mode. The I²C bus is enabled for use by other devices on the network while in idle mode.

Register List

This device is controlled and configured via a number of on-chip registers, The table below lists the registers and their access. All address locations are 8 bits.

Pin Description

Pin	Name	Type	Function
1	GND	G	Ground
2	VIN	P	Supply Voltage-2.7 to 6.5 VDC (module is regulated to 2.5DC)
3	DRDY	I	Data Ready, interrupt. (internally pulled high, see datasheet for details)
4	SCL	I	I²C Clock (Pulled high on module, Clock 160Hz Default)
5	SDA	IO	I²C Data (Pulled high on module)

Working Principle

Earth's magnetic field ise present in space which points towards the magnetic north as shown in below image. Current carrying conductor also generates a magnetic field around itself. Hence, whenever a current carrying conductor is placed in space, it experiences the effect of the earth's magnetic field affecting the flow of the electrons through that conductor. These changes in the flow of the electrons are used for identifying the heading or direction of the magnetic field. This is the basic working principle of the magnetometer.

The correspondent movement of the nickel-iron material in space experiences earth's magnetic field which changes the material's resistance, and hence we get resultant voltage changes across the bridge. This change in voltages is used to get the magnetic field direction in space.The components of Earth Magnetic Field (Hx, Hy) are parallel to the earth's surface and are used in determining the compass direction. Only the X and Y components of the earth's field are used in determining the azimuth angle, or the compass direction.

With this, let's see some of the features of HMC5883L like,

- It can be used for low cost compassing and magnetometry.

- It has 12-bit ADC and compass heading accuracy is up to 1° to 2°.
- It has Honeywell's Anisotropic Magneto Resistive (AMR) technology which provides precision in axis sensitivity and linearity.
- It uses I2C communication protocol to communicate with microcontrollers.

HMC5883L module has five pins as shown in the figure given below,

- VCC: Connect 5V DC supply to this pin.
- GND: Connect ground to this pin.
- SCL: Connect serial clock out from master device to this pin.
- SDA: Connect serial data line from master device to this pin.
- DRDY: Data Ready status signal pin output from module to master device.

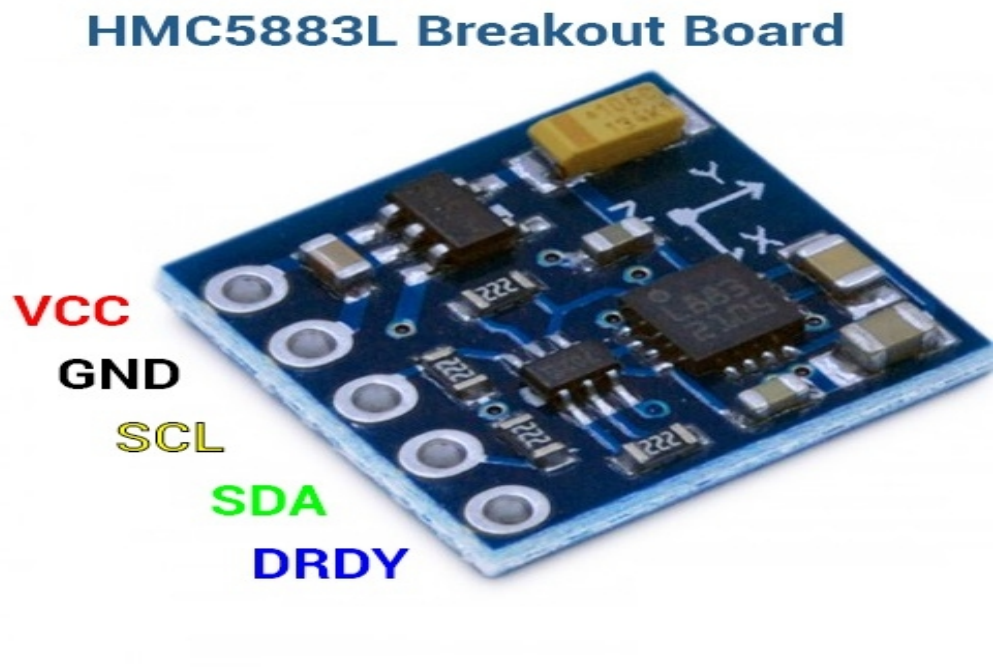


Fig 6.2 HMC5883l (3-axis compass)

Communicate with HMC5883L

HMC5883L uses I2C protocol for communication. HMC5883L act as a slave device. Its I2C device address is 0x1E. Its read and write operation addresses are:

- Slave device write address (SLA+W): 0x3C

- Slave device read address (SLA+R): 0x3D

Neo-6M (GPS)

NEO-6M GPS Module can track up to 22 satellites and identifies locations anywhere in the world. It may serve as a great launch pad for anyone looking to get into the world of GPS. They are low power (suitable for battery powered devices), inexpensive, easy to interface with and are insanely popular among hobbyists.

Hardware Overview of NEO-6M GPS Module

At the heart of the module is a NEO-6M GPS chip from u-blox. The chip measures less than the size of a postage stamp but packs a surprising amount of features into its little frame.It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current.Unlike other GPS modules, it can do up to 5 location updates a second with 2.5m Horizontal position accuracy. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a 0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART. The module supports baud rate from 4800bps to 230400bps with default baud of 9600.

How GPS works

GPS satellites circle the Earth twice a day in a precise orbit. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to decode and compute the precise location of the satellite. GPS receivers use

this information and trilateration to calculate a user's exact location. Essentially, the GPS receiver measures the distance to each satellite by the amount of time it takes to receive a transmitted signal. With distance measurements from a few more satellites, the receiver can determine a user's position and display it. To calculate your 2-D position (latitude and longitude) and track movement, a GPS receiver must be locked on to the signal of at least 3 satellites. With 4 or more satellites in view, the receiver can determine your 3-D position (latitude, longitude and altitude). Generally, a GPS receiver will track 8 or more satellites, but that depends on the time of day and where you are on the earth. Here are complete specifications:

Receiver Type	50 channels, GPS L1(1575.42Mhz)
Horizontal Position Accuracy	2.5m
Navigation Update Rate	1HZ (5Hz maximum)
Capture Time	Cool start: 27sHot start: 1s
Navigation Sensitivity	-161dBm
Communication Protocol	NMEA, UBX Binary, RTCM
Serial Baud Rate	4800-230400 (default 9600)
Operating Temperature	-40°C ~ 85°C
Operating Voltage	2.7V ~ 3.6V
Operating Current	45mA
TXD/RXD Impedance	510Ω

Position Fix LED Indicator

There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It'll blink at various rates depending on what state it's in:

- No Blinking – It's searching for satellites.

- Blink every 1s – Position Fix is found(The module can see enough satellites).

3.3V LDO Regulator

The operating voltage of the NEO-6M chip is from 2.7 to 3.6V. But the good news is that, the module comes with MIC5205 ultra-low dropout 3V3 regulator from MICREL. The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.

Battery & EEPROM

The module is equipped with an HK24C32 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I2C.
The module also contains a rechargeable button battery which acts as a super-capacitor. An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data(GNSS orbit data) and module configuration. But it's not meant for permanent data storage As the battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1s. This allows much faster position locks.
Without the battery the GPS always cold-start so the initial GPS lock takes more time. The battery is automatically charged when power is applied and maintains data for up to two weeks without power.

Antenna

An antenna is required to use the module for any kind of communication. So, the module comes with a patch antenna having -161dBm sensitivity. Patch antenna is great for most projects. But if you want to achieve more sensitivity or put your module inside a metal case, you can also snap on any 3V active GPS antenna via the U.FL connector.

Once your position has been determined, the GPS unit can calculate other information, such as:

- Speed

- Bearing
- Track
- Trip dist
- Distance to destination

What's the signal?

GPS satellites transmit at least 2 low-power radio signals. The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects, such as buildings and mountains. However, modern receivers are more sensitive and can usually track through houses.

A GPS signal contains 3 different types of information: Pseudo-random code is an I.D. code that identifies which satellite is transmitting information. You can see which satellites you are getting signals from on your device's satellite page.
Ephemeris data is needed to determine a satellite's position and gives important information about the health of a satellite, current date and time. Almanac data tells the GPS receiver where each GPS satellite should be at any time throughout the day and shows the orbital information for that satellite and every other satellite in the system.
The NEO-6M GPS module is shown in the figure below. It comes with an external antenna and does not come with header pins. So you will need to solder it.

Overview of NEO-6M GPS Module

The heart of the module is a NEO-6M GPS chip from u-blox. It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a "0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART. Pin out

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.
TxD (Transmitter) pin is used for serial communication.
RxD (Receiver) pin is used for serial communication.
VCC supplies power for the module. You can directly connect it to the 5V pin on the Arduino.

Fig 6.3 NEO-6M GPS Module

HC-05 (Bluetooth Module)

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. ... HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data. HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds. This module works on 3.3 V.

BLUETOOTH v2.1 BLUETOOTH 4.0 (LE)
Range Up to 100 m Up to 100 m
Max range (free field) Around 100 m (class 2 outdoors) Around 100 m (outdoors) Frequency 2.402 – 2.481 GHz 2.402 – 2.481

Serial Bluetooth module for Arduino and other microcontrollers

- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

How to Use the HC-05 Bluetooth module

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description. It is very easy to pair the HC-05 module with

microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the figure below

- Wireless communication between two microcontrollers
- Communicate with Laptop, Desktops and mobile phones
- Data Logging application
- Consumer applications
- Wireless Robots
- Home Automation

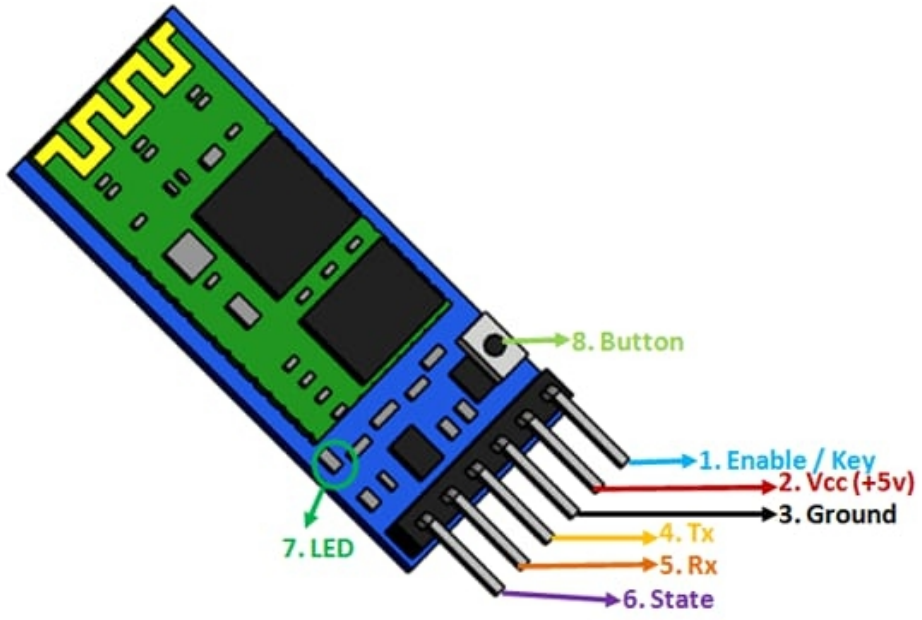


Fig 6.4 HC-05 Bluetooth module

Pin Number	Pin Name	Description
1	Enable/ Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TX - Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX - Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module ● Blink once in 2 sec: Module has entered Command Mode ● Repeated Blinking: Waiting for connection in Data Mode ● Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

HC-05 Default Setting

Default Bluetooth Name: “HC-05”

Default Password: 1234 or 0000

Default Communication: Slave

Default Mode: Data Mode

Data Mode Baud Rate: 9600, 8, N, 1

Command Mode Baud Rate: 38400, 8, N, 1

Default firmware: LINVOR

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

Where to use HC-05 Bluetooth Module

The HC-05 is a very cool module which can add two-way (full-duplex) wireless functionality to your projects. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available which makes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. We can also configure the default values of the module by using the command mode. So if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you. However do not expect this module to transfer multimedia like photos or songs; you might have to look into the CSR8645 module for that.

How to use HC-05 Bluetooth Module

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description. It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU.

Applications

1. Wireless communication between two microcontrollers
2. Communicate with Laptop, Desktops and mobile phones
3. Data Logging application
4. Consumer applications
5. Wireless Robots
6. Home Automation

L298N Motor Driver

If you are planning on assembling your new robot friend, you will eventually want to learn about controlling DC motors. One of the easiest and inexpensive way to control DC motors is to interface L298N Motor Driver with Arduino. It can control both speed and spinning direction of two DC motors.

Controlling a DC Motor

In order to have a complete control over DC motor, we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

- PWM – For controlling speed
- H-Bridge – For controlling rotation direction

PWM – For controlling speed

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) PWM is a technique where average value of the input voltage is adjusted by sending a series of ON-OFF pulses.

The average voltage is proportional to the width of the pulses known as Duty Cycle. 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).

H-Bridge – For controlling rotation direction

The DC motor's spinning direction can be controlled by changing polarity of its input voltage. A common technique for doing this is to use an H-Bridge. An H-Bridge circuit contains four switches with the motor at the center forming an H-like arrangement.

Closing two particular switches at the same time reverses the polarity of the voltage applied to the motor. This causes change in spinning direction of the motor.

L298N Motor Driver IC

At the heart of the module is the big, black chip with chunky heat sink is an L298N. The L298N is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors. That means it can individually drive up to two motors making it ideal for building two-wheel robot platforms.

Power Supply

The L298N motor driver module is powered through 3-pin 3.5mm-pitch screw terminals. It consists of pins for motor power supply(Vs), ground and 5V logic power supply (Vss). The module has an on-board 78M05 5V regulator from STMicroelectronics. It can be enabled or disabled through a jumper. When this jumper is in place, the 5V regulator is enabled, supplying logic power supply (Vss) from the motor power supply(Vs). In this case, 5V input terminal acts as an output pin and delivers 5V 0.5A. You can use it to power up the Arduino or other circuitry that requires 5V power supply.

When the jumper is removed, the 5V regulator gets disabled and we have to supply 5 Volts separately through 5 Volt input terminal.

Voltage Drop of L298N

The voltage drop of the L298N motor driver is about **2V**. This is due to the internal voltage drop in the switching transistors in the H-Bridge circuit.

So, if we connect 12V to the motor power supply terminal, the motors will receive voltage around 10V. This means that a 12V DC motor will never spin at its maximum speed. To get maximum speed out of motor, the motor power supply should be bit higher voltage(2V) than motor's actual voltage requirement. Considering the voltage drop of 2V, if you are using 5V motors you'll need to provide 7V at motor power supply terminal. If you have 12V motors then your motor supply voltage should be 14V.

Output Pins

The L298N motor driver's output channels for the motor A and B are broken out to the edge of the module with two 3.5mm-pitch screw terminals. You can connect two DC motors having voltages between 5 to 35V to these terminals. Each channel on the module can deliver up to 2A to the DC motor. However, the amount of current supplied to the motor depends on system's power supply.

Control Pins

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 viz. Direction control pins & Speed control pins.

Direction Control Pin

Using the direction control pins, we can control whether the motor spins forward or backward. These pins actually control the switches of the H-Bridge circuit inside L298N IC. The module has two direction control pins for each channel. The IN1 and IN2 pins control the spinning direction of the motor A while IN3 and IN4 control motor B. The spinning direction of a motor can be controlled by applying either a logic HIGH(5 Volts) or logic LOW(Ground) to these inputs. The below chart illustrates how this is done.

Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

Speed Control Pins

The speed control pins viz. ENA and ENB are used to turn the motors ON, OFF and control its speed. Pulling these pins HIGH will make the motors spin, pulling it LOW will make them stop. But, with Pulse Width Modulation (PWM), we can actually control the speed of the motors.

The module usually comes with a jumper on these pins. When this jumper is in place, the motor is enabled and spins at maximum speed. If you want to control the speed of motors programmatically, you need to remove the jumpers and connect them to PWM-enabled pins on Arduino.

L298N Motor Driver Module Pinout

Before diving into hookup and example code, let's first take a look at its Pinout.

- Vcc pin supplies power for the motor. It can be anywhere between 5 to 35V. Remember, if the 5V-EN jumper is in place, you need to supply 2 extra volts than motor's actual voltage requirement, in order to get maximum speed out of your motor.

- GND is a common ground pin.

- 5V pin supplies power for the switching logic circuitry inside L298N IC. If the 5V-EN jumper is in place, this pin acts as an output and can be used to power up your Arduino. If the 5V-EN jumper is removed, you need to connect it to the 5V pin on Arduino.

- ENA pins are used to control speed of Motor A. Pulling this pin HIGH(Keeping the jumper in place) will make the Motor A spin, pulling it LOW will make the motor stop. Removing the jumper and connecting this pin to PWM input will let us control the speed of Motor A.

- IN1 and IN2 pins are used to control spinning direction of Motor A. When one of them is HIGH and other is LOW, the Motor A will spin. If both the inputs are either HIGH or LOW the Motor A will stop.

- IN3 and IN4 pins are used to control spinning direction of Motor B. When one of them is HIGH and other is LOW, the Motor B will spin. If both the inputs are either HIGH or LOW the Motor B will stop.

- ENB pins are used to control speed of Motor B. Pulling this pin HIGH(Keeping the jumper in place) will make the Motor B spin, pulling it LOW will make the motor stop. Removing the jumper and connecting this pin to PWM input will let us control the speed of Motor B.

- OUT1 and OUT2 pins are connected to Motor A.

- OUT3 and OUT4 pins are connected to Motor B.

Wiring L298N motor driver module with Arduino UNO

Now that we know everything about the module, we can begin hooking it up to our Arduino! Start by connecting power supply to the motors. In our experiment we are using DC Gearbox Motors(also known as 'TT' motors) that are usually found in two-wheel-drive robots. They are rated for 3 to 12V. So, we will connect external 12V power supply to the VCC terminal. Considering internal voltage drop of L298N IC, the motors will receive 10V and will spin at slightly lower RPM. But, that's OK.

Next, we need to supply 5 Volts for the L298N's logic circuitry. We will make use of the on-board 5V regulator and derive the 5 volts from the motor power supply so, keep the 5V-EN jumper in place.

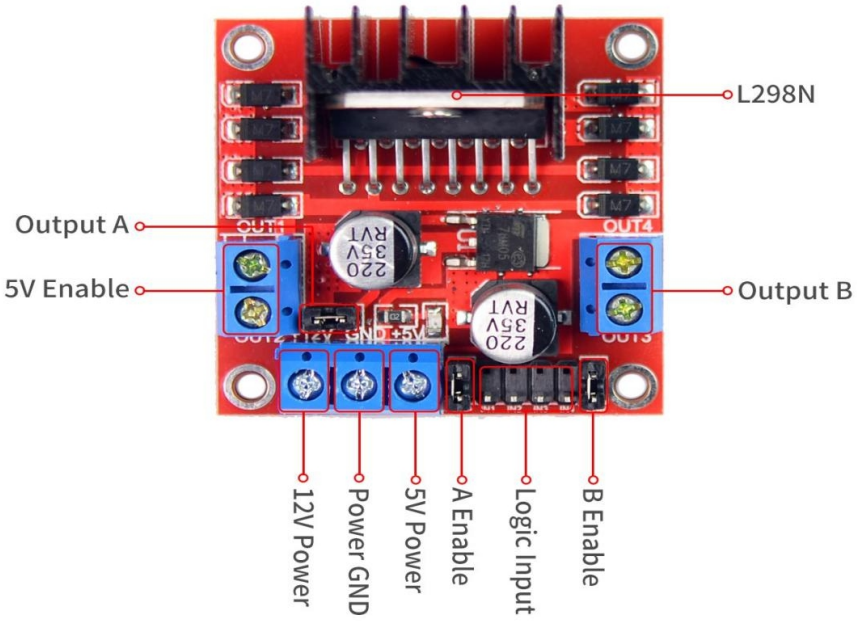


Fig 6.5 L298N Motor Driver

Now, the input and enable pins(ENA, IN1, IN2, IN3, IN4 and ENB) of the L298N module are connected to six Arduino digital output pins(9, 8, 7, 5, 4 and 3). Note that the Arduino output pins 9 and 3 are both PWM-enabled. Finally, connect one motor to terminal A(OUT1 & OUT2) and the other motor to terminal B(OUT3 & OUT4). You can interchange your motor's connections, technically, there is no right or wrong way.

Servo Motor

The servo motor is most commonly used for high technology devices in the industrial applications like automation technology. It is a self contained electrical device, that rotates parts of machine with high efficiency and great precision. Moreover the output shaft of this motor can be moved to a particular angle. Servo motors are mainly used in home electronics, toys, cars, airplanes and many more devices. Thus this blog discusses the definition, types, mechanism, principle, working, controlling, and lastly the applications of a servo machine.

Definition :

A servo motor is a rotary actuator or a motor that allows for a precise control

in terms of the angular position, acceleration, and velocity. Basically it has certain capabilities that a regular motor does not have. Consequently it makes use of a regular motor and pairs it with a sensor for position feedback .

Types of servo motors :

Servo motors can be of different types on the basis of their applications. The most important amongst them are : AC servo motor, DC servo motor, brushless DC servo motor, positional rotation servo motor, continuous rotation servo motor, and linear servo motor. A typical servo motor comprises of three wires namely- power, control, and ground. The shape and size of these motors depends on their applications.

1. DC servo motor :

The basic operating principle of DC motor is the same as other electromagnetic motors. The design, construction, and the modes of operation are different. The rotors of this kind of motor are designed with long rotor length and smaller diameters. Their size is larger than that of conventional motors of same power ratings. There are various types of dc servo motors which are :

1. Series motors :

The series motors have a high starting torque and draws large current. The speed regulation of this kind of motor is poor.

2. Split series motor :

They are the motors with split-field rate with some fractional kilowatts. Split series motor has a typical torque-speed curve. This curve denotes high stall torque and a rapid reduction in torque with high speed.

3. Shunt control motor :

It has two separate windings:

1.field winding – on the stator.

2.armature winding – on the rotor of the machine.

Both windings are connected to a dc supply source.

4. Permanent magnet shunt motor :

It is a fix excitation motor where the field is actually supply by a permanent

magnet. Furthermore, the performance is similar to armature controlled fixed field motor.

2. AC servo motor :

AC servomotors are AC motors in which incorporate encoders are use with controllers for providing feedback and close-loop control. Hence, these motors can be positioned to high accuracy. Thus they can be controlled exactly as per requirement for the application. The classification of AC servomotors is done into two types. These are 2 phase and 3 phase AC servo motor. Now most of the AC servomotors are of the two-phase squirrel cage induction motor type. They are used for low power applications. Furthermore the three phase squirrel cage induction motor is now utilized for applications where high power system are in use.

3. Brushless DC servomotor :

BLDC motors are also commonly known as electronically commutated motors or synchronous motors powered by DC electricity via inverter or switching power supply. Hence this provides an AC electric current to drive each phase of motor via a closed loop controller. The controller provides pulses of current to the motor windings that control the speed and torque of the motor. The construction of a brushless motor system is typically similar to a permanent magnet synchronous motor. Finally the advantages of the brushless motor over brushed motors are high power to weight ratio, high speed, and electronic control. The brushless motors find applications in such places as computer peripherals (disk drives, printers), hand-held power tools, and vehicles ranging from model aircrafts to automobiles.

4. Positional rotation servo motor :

Positional rotation servo motor is the most important servo motor. Hence it is also the most common type of servo motor. The shaft output rotates in about 180 degree. Additionally it includes physical stops located in gear mechanism to stop turning outside these limits to guard the rotation sensor. These common servos involve in radio controlled water, ratio controlled cars, aircraft, robots, toys and many other applications.

5. Continuous rotation servo motor :

Continuous rotation servo motor relates to the common positional rotation servo motor, but it can go in any direction indefinitely. The control signal, rather than setting the static position of the servo, is understood as speed and direction of rotation. The range of potential commands sources the servo to rotate clockwise or anticlockwise as preferred, at changing on the command signal. Thus this type of motor is used in a radar dish if you are riding, one on a robot or you can use one as a drive motor on a mobile robot.

6. Linear servo motor :

Linear servo motor is also similar to the positional rotation servo motor discussed above, but with extra gears to alter the output from circular to back and forth. Although these servo motors are not likely to be found, but sometimes you can find them at hobby stores where they are used as actuators in higher model airplanes .

Principle of working :

Servo motor works on the PWM (Pulse Width Modulation) principle, which means its angle of rotation is controlled by the duration of pulse applied to its control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears.

Mechanism of servomotor :

Basically a servo motor is a closed-loop servomechanism that uses position feedback to control its motion and final position. Moreover the input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft . The motor is incorporates some type of encoder to provide position and speed feedback. In the simplest case, we measure only the position. Then the measured position of the output is compared with the command position, the external input to controller. Now If the output position differs from that of the expected output, an error signal generates. Which then causes the motor to rotate in either direction, as per need to bring the output shaft to the appropriate position. As the position approaches, the error signal reduces to zero. Finally the motor stops.

The very simple servomotors can position only sensing via a potentiometer

and bang-bang control of their motor. Further the motor always rotates at full speed. Though this type of servomotor doesn't have many uses in industrial motion control, however it forms the basis of simple and cheap servo used for radio control models. Servomotors also find uses in optical rotary encoders to measure the speed of output shaft and a variable-speed drive to control the motor speed. Now this, when combined with a PID control algorithm further allows the servomotor to be in its command position more quickly and more precisely with less overshooting .

Working of servomotors :

Servo motors control position and speed very precisely. Now a potentiometer can sense the mechanical position of the shaft. Hence it couples with the motor shaft through gears. The current position of the shaft is converted into electrical signal by potentiometer, and is compared with the command input signal. In modern servo motors, electronic encoders or sensors sense the position of the shaft . We give command input according to the position of shaft . If the feedback signal differs from the given input, an error signal alerts the user. We amplify this error signal and apply as the input to the motor, hence the motor rotates. And when the shaft reaches to the require position , error signal become zero , and hence the motor stays standstill holding the position.

The command input is in form of electrical pulses . As the actual input to the motor is the difference between feedback signal (current position) and required signal, hence speed of the motor is proportional to the difference between the current position and required position . The amount of power require by the motor is proportional to the distance it needs to travel .

Controlling of servomotors :

Usually a servomotor turns 90 degree in either direction hence maximum movement can be 180 degree . However a normal servo motor cannot rotate any further to a build in mechanical stop.

We take three wires are out of a servo : positive , ground and control wire. A servo motor is control by sending a pulse width modulated(PWM)

signal through the control wire . A pulse is sent every 20 milliseconds. Width of the pulses determine the position of the shaft .

for example ,

A pulse of 1ms will move the shaft anticlockwise at -90 degree , a pulse of 1.5ms will move the shaft at the neutral position that is 0 degree and a pulse of 2ms will move shaft clockwise at +90 degree.

When we command a servo motor to move by applying pulse of appropriate width, the shaft moves to and holds the require position of the shaft. However the motor resists to change . Pulses need repetition for the motor to hold the position .

Applications :

1. Robotics : At every joint of the robot, we connect a servomotor. Thus giving the robot arm its precise angle.
2. Conveyor belts : servo motors move , stop , and start conveyor belts carrying product along to various stages , for example , in product packaging/ bottling, and labelling .
3. Camera auto focus : A highly precise servo motor build into the camera corrects a camera lens to sharpen out of focus images.

4.Solar tracking system : Servo motors adjust the angle of solar panels throughout the day and hence each panel continues to face the sun which results in harnessing maximum energy from sunup to sundown .

Fig 6.6 Servo motor

DC Motor

A Direct Current (DC) motor is a rotating electrical device that converts direct current, of electrical energy, into mechanical energy. An Inductor (coil) inside the DC motor produces a magnetic field that creates rotary motion as DC voltage is applied to its terminal. Inside the motor is an iron shaft, wrapped in a coil of wire. This shaft contains two fixed, North and South, magnets on both sides which causes both a repulsive and attractive force, in turn, producing torque. ISL Products designs and manufactures both brushed DC motors and brushless DC motors. We tailor our DC motors size and performance to meet your desired specs.

DC Gear Motor

A gear motor is an all-in-one combination of a motor and gearbox. The addition of a gear head to a motor reduces the speed while increasing the torque output. The most important parameters in regards to gear motors are speed (rpm), torque (lb-in) and efficiency (%). In order to select the most suitable gear motor for your application you must first compute the load, speed and torque requirements for your application. ISL Products offers a variety of Spur Gear Motors, Planetary Gear Motors and Worm Gear Motors to meet all application requirements. Most of our DC motors can be complimented with one of our unique gearheads, providing you with a highly efficient gear motor solution.

Fig 6.7 DC Gear Motor

AS 608- Optical Fingerprint Sensor

The working principle of the fingerprint sensor mainly depends on the processing. The fingerprint processing mainly includes two elements namely enrollment and matching. In fingerprint enrolling, every user requires to place the finger twice.So that the system will check the finger images to process as well as to generate a pattern of the finger and it will be stored. When matching, a user places the finger using an optical sensor then the system will produce a pattern of the finger & compares it with the finger library templates. The system will evaluate the exits finger with a precise pattern which is selected within the module. Similarly, for 1: N matching, the scanning system will look for the complete finger records for the finger matching. In both situations, the scanning system will go back to the corresponding result, success otherwise crash.

- Supply voltage: 3.6 - 6.0VDC
- Operating current: 120mA max
- Peak current: 150mA
- max Fingerprint imaging time: <0.001% (Security level 3) False Reject Rate: <1.0% (Security level 3)
- Interface: TTL Serial Baud rate: 9600, 19200, 28800, 38400, 57600 (default is 57600)
- Working temperature rating: -20C to +50C
- Working humidy: 40%-85%
- RH Full Dimensions: 56 x 20 x 21.5mm
- Exposed Dimensions (when placed in box): 21mm x 21mm x 21mm triangular Weight: 20 grams

AS608 Optical Fingerprint Sensor Pinout

- This Module has 4 useful pins:
- V+: Module power supply – 3.3V
- GND: Ground
- TX: Serial Transmitter
- RX: Serial Receiver

Enrolling vs. Searching

There are basically two requirements for using the optical fingerprint sensor. First is you'll need to enroll fingerprints - that means assigning ID #'s to each print so you can query them later. Once you've enrolled all your prints, you can easily 'search' the sensor, asking it to identify which ID (if any) is currently being photographed. You can enroll using the Windows software (easiest and neat because it shows you the photograph of the print) or with the Arduino sketch (good for when you don't have a Windows machine handy or for on-theroad enrolling).

Specifications

The specifications of this sensor include the following.

- The fingerprint sensor is an optical type
- The interface is USB1.1/ TTL logical level (UART)
- The speed of scanning is 0.5 sec
- The speed of verification is 0.3 sec
- The capacity storage is 1000
- The security level is 5
- The baud rate of RS232 is 4800BPS ~115200BPS variable
- Current is typical 50 mA, and peak 80mA
- The corresponding technique is 1: N
- Fixed indicators-15KV bright green backlight
- The life of the sensor is 100 million times
- The dimension is 44.1 X 20 X 23.5mm
- The size of the character file is 256 bytes
- The template size is 512 bytes
- The FRR (False Rejection Rate) is <1.0%
- The FAR (False Acceptance Rate) is 0.001%
- Voltage is 4.2 to 6.0 VDC

- Operating surroundings temperature is -20° C to 40°

Fig 6.8 AS608 Optical Fingerprint Sensor

PIEZO BUZZERS

- wide variety of available sizes / frequencies / sound outputs:
  - self-oscillating buzzers with signal generators
  - buzzers without signal generators
  - multi-tone sound generators /sirens
- low cost / high sound outputs
- can be mounted in printed circuit boards
- competitive pricing
- fast delivery

What Is A Piezo Buzzer?

In simplest terms, a piezo buzzer is a type of electronic device that's used to produce a tone, alarm or sound. It's lightweight with a simple construction, and it's typically a low-cost product. Yet at the same time, depending on the piezo ceramic buzzer specifications, it's also reliable and can be constructed in a wide range of sizes that work across varying frequencies to produce different sound outputs.

For instance, at APC International, Ltd., we offer piezo buzzers without signal generators, self-oscillating buzzers that have signal generators and even multi-tone sound generators — often used in alarms and sirens. Regardless of the model you choose, our piezo buzzers offer high sound outputs. Plus, since they can be mounted on circuit boards, they're highly useful in a wide range of applications and assemblies.

Despite different construction methods that affect the cost of piezo buzzers, all of our prices are highly competitive. In addition, thanks to our state-of-the-art production facility, our delivery times are some of the fastest in the industry.

How to Use Piezoelectric Buzzers

The use of the piezo ceramic buzzer was discovered thanks to an inversion of the piezoelectricity principle that was discovered by Jacques and Pierre Curie back in 1880. They found that electricity could be generated when a mechanical pressure was applied to particular materials — and the inverse was true as well. So when certain piezoelectric materials are subjected to an alternating field of electricity, the piezo buzzer element — often a manmade piezoceramic material — stretches and compresses in sequence with the frequency of the current. As a result, it produces an audible sound.

Unlike magnetic buzzers that have a narrow operating voltage of somewhere between one and 16 volts, piezo buzzers can typically operate anywhere between three and 250 volts. In addition, magnetic buzzers have a higher power consumption of 30 to 100 milliamperes, while piezo buzzers normally consume less than 30 milliamperes — even at higher rate frequencies. And although piezo buzzers require a larger footprint than magnetic buzzers, they produce a higher sound pressure level.

Typical Applications of a Piezo Buzzer

Thanks to both the reliability and flexibility of piezoelectric vibration plates to produce audible signals — ranging from monotone buzzes and alarms to multi-tones and melodies — their applications in small, high-density assemblies are wide-ranging. What's more: Their low power consumption makes them ideal for many battery-operated devices. With such characteristics, piezo buzzers are regularly used in alarms, warning devices and automobile

alerts. In addition, since they can produce a wide range of audible signals, they're also used in pest deterrent devices. And in the consumer electronics field, some of their most popular applications include sound generators in computers, telephones, toys and games — to name just a few.

Fig 6.9 Piezo buzzer

Bread Board

Features and Specifications

- 2 Distribution Strips, 200 tie-points
- 630 tie-points in IC/ circuit areas
- ABS plastic with color legend
- Dimension: 6.5*4.4*0.3 inch
- Hole/Pitch Style: Square wire holes (2.54mm)
- ABS heat Distortion Temperature: 84° C (183° F)
- Rating: 300/3 to 5Amps
- Insulation Resistance : 500MΩ / DC500V

- Withstanding Voltage : 1,000V AC / 1 minute
- Insertion Wire Size: 21 to 26 AWG wire

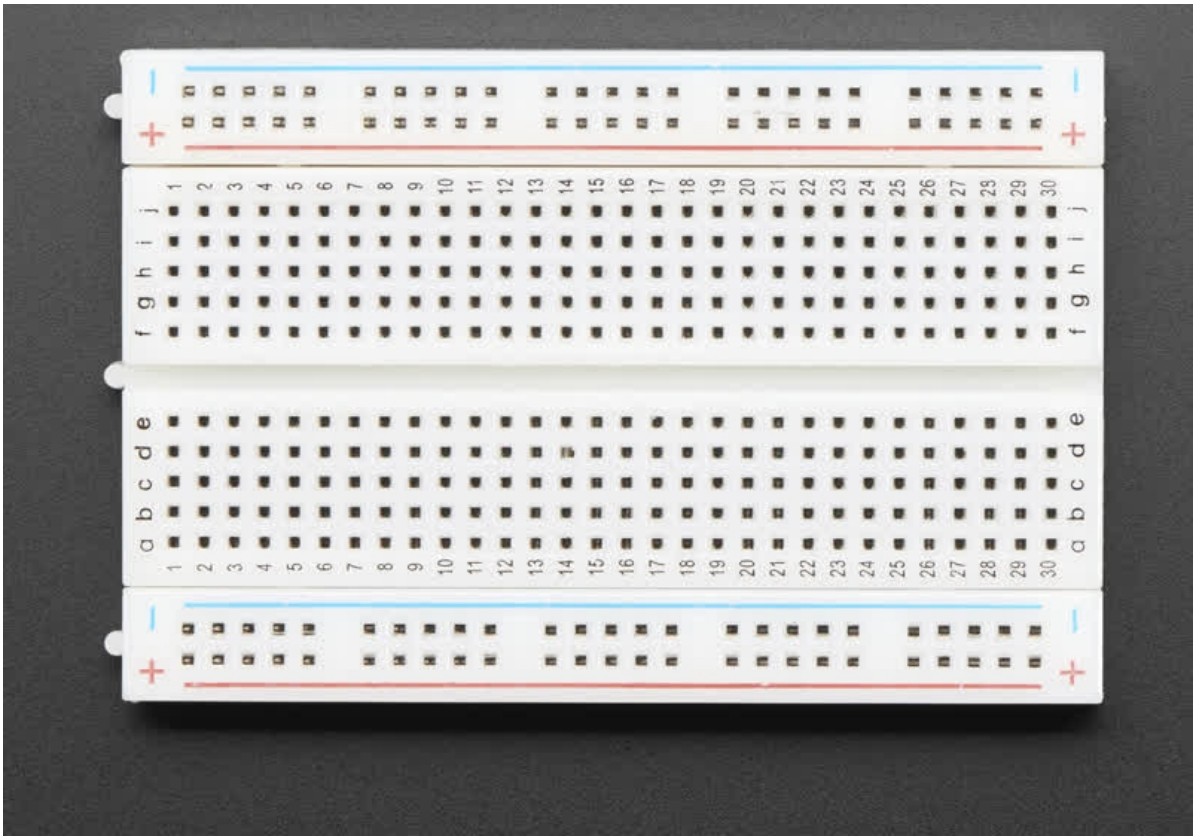


Fig 6.10 Bread board

Brief about Bread Board

Breadboard is a plastic board with a bunch of tiny holes and is used for building and testing circuits. It has holes on them which are connected internally in a particular pattern as shown in the below picture. The holes which are connected through green line represents they are connected internally. The Red line indicates Power, which is normally connected to the power rail. The Blue line indicates Ground, which is normally connected to the ground of the circuit. IC's like Decade Counter can be placed in the middle breadboard to share the 1st eight pins to the yellow line and the 2nd eight pins to the green lines.

Jumper Wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.

What Do the Colors Mean?

Though jumper wires come in a variety of colors, the colors don't actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power.

Fig 6.11 Jumper wires

Source Code

```
//scootfree with normal motors tiny gps
// Imports
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#include <MechaQMC5883.h>
#include "./TinyGPS.h"
//#include "./Definitions.h"
MechaQMC5883 qmc;
// Pin variables
#define GPS_TX_PIN 6
#define BUZZER 3
#define READ_BLUE 2
//#define PIN_SPEED_RIGHT 9
//#define PIN_SPEED_LEFT 5
#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
#define BLUETOOTH_TX_PIN 10
#define BLUETOOTH_RX_PIN 11
// Bluetooth GPS input
#define SOP '<'
#define EOP '>'
// Motor stuffs
#define RC_NEUTRAL 1500
#define RC_MAX 1600
#define RC_MIN 1400
```

```
// If one motor tends to spin faster than the other, add offset
#define MOTOR_A_OFFSET 0 //0
#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 -1° 14'
#define DECLINATION_ANGLE 0.237258f
// 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 0.35f //0
#define GPS_STREAM_TIMEOUT 18
// 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 waypoints
// Keeps the robot from driving away if there is a problem
#define GPS_WAYPOINT_TIMEOUT 25
// Struct to combine our coordinates into one struct for ease of use
struct GeoLoc {
  float lat;
  float lon;
};
//
// GPS
TinyGPS gps;
//#define GPS_BAUD 4800
// Motors
//Servo servoThrottleRight;
//Servo servoThrottleLeft;
```

```
// Master Enable
bool enabled = true;
// Bluetooth input
bool started = false;
bool ended = false;
char inData[80]; // creates an 80 character array called "inData"
byte index; //creates a variable type=byte called "index"
double Long; //variable for longitude coordinate
double Lat; //variable for latitude coordinate
int val = 0;
// Serial components
SoftwareSerial
bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);
SoftwareSerial nss(GPS_TX_PIN,255); // TXD to digital pin 6
/* Compass */
Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

GeoLoc checkGPS() {
 Serial.println("Reading onboard GPS: ");
 bool newdata = false;
 unsigned long start = millis();
 while (millis() - start < GPS_UPDATE_INTERVAL) {
  if (feedgps())
    newdata = true;
 }
 if (newdata) {
  return gpsdump(gps);
 }
 GeoLoc robotLoc;
 robotLoc.lat = 0.0;
 robotLoc.lon = 0.0;
 return robotLoc;
}
```

```
// Get and process GPS data
GeoLoc gpsdump(TinyGPS &gps) {
  float flat, flon;
  unsigned long age;
  gps.f_get_position(&flat, &flon, &age);
  GeoLoc robotLoc;
  robotLoc.lat = flat;
  robotLoc.lon = flon;
Serial.print("Robot Latitude: ");Serial.print(robotLoc.lat, 7); Serial.print(",
Longitude: "); Serial.println(robotLoc.lon, 7);
  return robotLoc;
}
// Feed data as it becomes available
bool feedgps() {
  while (nss.available()) {
    if (gps.encode(nss.read()))
      return true;
  }
  return false;
}
void displayCompassDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print   ("Max Value”); Serial.print(sensor.max_value); Serial.println("
uT");
  Serial.print   ("Min Value:"); Serial.print(sensor.min_value); Serial.println("
uT");
  Serial.print    ("Resolution:"); Serial.print(sensor.resolution); Serial.println("
uT");
```

```
  Serial.println("-----------------------------------");
  Serial.println("");
  delay(500);
}
#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.295779513082320876f
#endif
float geoBearing(struct GeoLoc &a, struct GeoLoc &b) {
  float y = sin(b.lon-a.lon) * cos(b.lat);
  float x = cos(a.lat)*sin(b.lat) - sin(a.lat)*cos(b.lat)*cos(b.lon-a.lon);
  return atan2(y, x) * RADTODEG;
}
  float geoDistance(struct GeoLoc &a, struct GeoLoc &b) {
  const float R = 6371000; // radius of earth in metres
  float p1 = a.lat * DEGTORAD;
  float p2 = b.lat * DEGTORAD;
  float dp = (b.lat-a.lat) * DEGTORAD;
  float dl = (b.lon-a.lon) * 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));
  // returns distance in meters
  return R * y;
}
float geoHeading() {
  /* Get a new sensor event */
  sensors_event_t event;
  //Serial.println("Inside geo heading after sensor");//<<<<<<testing inside geo
heading<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  //mag.getEvent(&event);
  int x,y,z;
  qmc.read(&x,&y,&z);
  //Serial.println("Inside geo heading after qmc");
```

```
 // 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(event.magnetic.y, event.magnetic.x);
  float heading = atan2(y, x);
  //Serial.println("Inside geo heading after atan2");
  // 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) {
  //servoThrottleRight.writeMicroseconds(speed + MOTOR_A_OFFSET);
   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);
  //digitalWrite(MOTOR_A_EN_PIN, HIGH);
}
void setSpeedMotorB(int speed) {
  //servoThrottleLeft.writeMicroseconds(speed + MOTOR_B_OFFSET);
```

```
  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);
  //digitalWrite(MOTOR_B_EN_PIN, HIGH);
}
void stop() {
  // stop the motors
  //servoThrottleRight.writeMicroseconds(RC_NEUTRAL);
  //servoThrottleLeft.writeMicroseconds(RC_NEUTRAL);
  // now turn off motors
//  digitalWrite(MOTOR_A_EN_PIN, LOW);
//  digitalWrite(MOTOR_A_EN_PIN, LOW);
  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;
  // Slowing down??? Needed???
  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) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MAX)) * autoSteerA);
////    int speedB = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MIN)) * autoSteerB);
   //int speedA = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MIN)) * autoSteerA);
   //int speedB = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MAX)) * 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 robotLoc = checkGPS();
  bluetoothSerial.listen();
  if (robotLoc.lat != 0 && robotLoc.lon != 0 && enabled) {
    float distance = 0;
    //Start move loop here
```

```
    do {
      //Serial.println("Inside                                                    do
while");//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      nss.listen();
      robotLoc = checkGPS();
      bluetoothSerial.listen();
      distance = geoDistance(robotLoc, loc);
 //Serial.println("Before calling the heading function");
      float heading = geoHeading();//<<<<<<<<<< testing here
      //
      //Serial.println("Heading:");Serial.println(heading);
      float bearing = geoBearing(robotLoc, loc) - geoHeading();
      Serial.print("Distance: ");
      Serial.println(distance);
      Serial.print("Bearing: ");
      Serial.println(geoBearing(robotLoc, loc));
      Serial.print("Heading: ");
      Serial.println(geoHeading())
      drive(distance, bearing);
      timeout -= 1;
    } while (distance > 3.0 && enabled && timeout>0);
    stop();
  }
}
void setupCompass() {
   /* Initialise the compass */
  if(!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections
*/
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }
```

```
  /* Display some basic information on this sensor */
  displayCompassDetails();
}
void chechBlue()
{
   //Serial.println(digitalRead(READ_BLUE));
  if(!digitalRead(READ_BLUE))
    {digitalWrite(BUZZER,HIGH);
    //Serial.println("Blue tooth not connected");
    }
    else{
    digitalWrite(BUZZER,LOW);
    //Serial.println("Blue tooth connected");
    }
  }
void setup()
{
  // Attaching motors
  //servoThrottleRight.attach(PIN_SPEED_RIGHT);
  //servoThrottleLeft.attach(PIN_SPEED_LEFT);
  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(BUZZER, OUTPUT);
  pinMode(READ_BLUE, INPUT);
 pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  //Debugging via serial
  Serial.begin(4800); //4800
```

```
  qmc.init();
  // Compass
  setupCompass();
  //GPS
  nss.begin(9600);
  //Bluetooth
  bluetoothSerial.begin(9600);
  //Blynk.begin(bluetoothSerial, auth);
}
// Testing
void testDriveNorth() {
  float heading = geoHeading();
  int testDist = 5;
  Serial.println(heading);
  while(!(heading < 5 && heading > -5)) {
    drive(testDist, heading);
    heading = geoHeading();
    Serial.println(heading);
    delay(500);
  }
  stop();
}
void loop()
{
  //Blynk.run();
  // Read all serial data available, as fast as possible
  chechBlue();
  while (bluetoothSerial.available() > 0)
  {
    char inChar = ((byte)bluetoothSerial.read());
    if (inChar == SOP)
    {
      index = 0;
      inData[index] = '\0';
```

```
    started = true;
    ended = false;
   }
   else if (inChar == EOP)
   {
    ended = true;
    break;
   }
   else
   {
    if (index < 79)
    {
     inData[index] = inChar;
     index++;
     inData[index] = '\0';
    }
   }
}
// We are here either because all pending serial
// data has been read OR because an end of
// packet marker arrived. Which is it?
if (started && ended)
{
  char *token = strtok(inData, ",");
  boolean first = true;
  while (token)
  {
    double val = atof(token);
    token = strtok(NULL, ",");
    if (first){
      Lat = val;
    }else{
      Long = val;
    }
```

```
      first = false;
    }
    // Reset for the next packet
    started = false;
    ended = false;
    index = 0;
    inData[index] = '\0';
    GeoLoc phoneLoc;
    phoneLoc.lat = Lat;
    phoneLoc.lon = Long;
  Serial.print("Phone  Latitude:  ");Serial.print(phoneLoc.lat,  7);  Serial.print(",
Longitude: ");
      Serial.println(phoneLoc.lon, 7);
    driveTo(phoneLoc, GPS_WAYPOINT_TIMEOUT);
  }
}
```

Tiny GPS.cpp Library

```
#include "Arduino.h"
#include "TinyGPS.h"
#define _GPRMC_TERM   "GPRMC"
#define _GPGGA_TERM   "GPGGA"
TinyGPS::TinyGPS()
:  _time(GPS_INVALID_TIME)
,  _date(GPS_INVALID_DATE)
,  _latitude(GPS_INVALID_ANGLE)
,  _longitude(GPS_INVALID_ANGLE)
,  _altitude(GPS_INVALID_ALTITUDE)
,  _speed(GPS_INVALID_SPEED)
,  _course(GPS_INVALID_ANGLE)
,  _last_time_fix(GPS_INVALID_FIX_TIME)
,  _last_position_fix(GPS_INVALID_FIX_TIME)
,  _parity(0)
```

```
,  _is_checksum_term(false)
,  _sentence_type(_GPS_SENTENCE_OTHER)
,  _term_number(0)
,  _term_offset(0)
,  _gps_data_good(false)
#ifndef _GPS_NO_STATS
,  _encoded_characters(0)
,  _good_sentences(0)
,  _failed_checksum(0)
#endif
{
  _term[0] = '\0';
}
//
// public methods
//
bool TinyGPS::encode(char c)
{
  bool valid_sentence = false;
  ++_encoded_characters;
  switch(c)
  {
  case ',': // term terminators
    _parity ^= c;
  case '\r':
  case '\n':
  case '*':
    if (_term_offset < sizeof(_term))
    {
      _term[_term_offset] = 0;
      valid_sentence = term_complete();
    }
    ++_term_number;
    _term_offset = 0;
```

```
    _is_checksum_term = c == '*';
    return valid_sentence;
  case '$': // sentence begin
    _term_number = _term_offset = 0;
    _parity = 0;
    _sentence_type = _GPS_SENTENCE_OTHER;
    _is_checksum_term = false;
    _gps_data_good = false;
    return valid_sentence;
  }
// ordinary characters
  if (_term_offset < sizeof(_term) - 1)
    _term[_term_offset++] = c;
  if (!_is_checksum_term)
    _parity ^= c;
  return valid_sentence;
}
#ifndef _GPS_NO_STATS
void TinyGPS::stats(unsigned long *chars, unsigned short *sentences,
unsigned short *failed_cs)
{
  if (chars) *chars = _encoded_characters;
  if (sentences) *sentences = _good_sentences;
  if (failed_cs) *failed_cs = _failed_checksum;
}
#endif
//
// internal utilities
//
int TinyGPS::from_hex(char a)
{
  if (a >= 'A' && a <= 'F')
    return a - 'A' + 10;
  else if (a >= 'a' && a <= 'f')
```

```
    return a - 'a' + 10;
  else
    return a - '0';
}
unsigned long TinyGPS::parse_decimal()
{
  char *p = _term;
  bool isneg = *p == '-';
  if (isneg) ++p;
  unsigned long ret = 100UL * gpsatol(p);
  while (gpsisdigit(*p)) ++p;
  if (*p == '.')
  {
    if (gpsisdigit(p[1]))
    {
      ret += 10 * (p[1] - '0');
      if (gpsisdigit(p[2]))
        ret += p[2] - '0';
    }
  }
  return isneg ? -ret : ret;
}
unsigned long TinyGPS::parse_degrees()
{
  char *p;
  unsigned long left = gpsatol(_term);
  unsigned long tenk_minutes = (left % 100UL) * 10000UL;
  for (p=_term; gpsisdigit(*p); ++p);
  if (*p == '.')
  {
    unsigned long mult = 1000;
    while (gpsisdigit(*++p))
    {
      tenk_minutes += mult * (*p - '0');
```

```
      mult /= 10;
    }
  }
return (left / 100) * 100000 + tenk_minutes / 6;
}
// Processes a just-completed term
// Returns true if new sentence has just passed checksum test and is validated
bool TinyGPS::term_complete()
{
  if (_is_checksum_term)
  {
    byte checksum = 16 * from_hex(_term[0]) + from_hex(_term[1]);
    if (checksum == _parity)
    {
      if (_gps_data_good)
      {
#ifndef _GPS_NO_STATS
        ++_good_sentences;
#endif
        _last_time_fix = _new_time_fix;
        _last_position_fix = _new_position_fix;
        switch(_sentence_type)
        {
        case _GPS_SENTENCE_GPRMC:
          _time      = _new_time;
          _date      = _new_date;
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          _speed     = _new_speed;
          _course    = _new_course;
          break;
        case _GPS_SENTENCE_GPGGA:
          _altitude  = _new_altitude;
          _time      = _new_time;
```

```
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          break;
        }
        return true;
      }
    }
#ifndef _GPS_NO_STATS
    else
      ++_failed_checksum;
#endif
    return false;
  }
  // the first term determines the sentence type
  if (_term_number == 0)
  {
    if (!gpsstrcmp(_term, _GPRMC_TERM))
      _sentence_type = _GPS_SENTENCE_GPRMC;
    else if (!gpsstrcmp(_term, _GPGGA_TERM))
      _sentence_type = _GPS_SENTENCE_GPGGA;
    else
      _sentence_type = _GPS_SENTENCE_OTHER;
    return false;
  }
  if (_sentence_type != _GPS_SENTENCE_OTHER && _term[0])
  switch((_sentence_type == _GPS_SENTENCE_GPGGA ? 200 : 100) +
_term_number)
  {
    case 101: // Time in both sentences
    case 201:
      _new_time = parse_decimal();
 _new_time_fix = millis();
    break;
```

```
    case 102: // GPRMC validity
      _gps_data_good = _term[0] == 'A';
      break;
    case 103: // Latitude
    case 202:
      _new_latitude = parse_degrees();
      _new_position_fix = millis();
      break;
    case 104: // N/S
    case 203:
      if (_term[0] == 'S')
        _new_latitude = -_new_latitude;
      break;
    case 105: // Longitude
    case 204:
      _new_longitude = parse_degrees();
      break;
    case 106: // E/W
    case 205:
      if (_term[0] == 'W')
        _new_longitude = -_new_longitude;
      break;
    case 107: // Speed (GPRMC)
      _new_speed = parse_decimal();
      break;
    case 108: // Course (GPRMC)
      _new_course = parse_decimal();
      break;
    case 109: // Date (GPRMC)
      _new_date = gpsatol(_term);
      break;
  case 206: // Fix data (GPGGA)
      _gps_data_good = _term[0] > '0';
     break;
```

```
    case 209: // Altitude (GPGGA)
      _new_altitude = parse_decimal();
      break;
    }
  return false;
}
long TinyGPS::gpsatol(const char *str)
{
  long ret = 0;
  while (gpsisdigit(*str))
    ret = 10 * ret + *str++ - '0';
  return ret;
}
int TinyGPS::gpsstrcmp(const char *str1, const char *str2)
{
  while (*str1 && *str1 == *str2)
    ++str1, ++str2;
  return *str1;
}
/* static */
float TinyGPS::distance_between (float lat1, float long1, float lat2, float long2)
{
  // returns distance in meters between two positions, both specified
  // as signed decimal-degrees latitude and longitude. Uses great-circle
  // distance computation for hypothetical sphere of radius 6372795 meters.
  // Because Earth is no exact sphere, rounding errors may be up to 0.5%.
 // Courtesy of Maarten Lamers
  float delta = radians(long1-long2);
  float sdlong = sin(delta);
 float cdlong = cos(delta);
  lat1 = radians(lat1);
  lat2 = radians(lat2);
  float slat1 = sin(lat1);
  float clat1 = cos(lat1);
```

```
  float slat2 = sin(lat2);
  float clat2 = cos(lat2);
  delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
  delta = sq(delta);
  delta += sq(clat2 * sdlong);
  delta = sqrt(delta);
  float denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
  delta = atan2(delta, denom);
  return delta * 6372795;
}
```

Tiny GPS.h Library

```
#ifndef TinyGPS_h
#define TinyGPS_h
#include "Arduino.h"
#define _GPS_VERSION 10 // software version of this library
#define _GPS_MPH_PER_KNOT 1.15077945
#define _GPS_MPS_PER_KNOT 0.51444444
#define _GPS_KMPH_PER_KNOT 1.852
#define _GPS_MILES_PER_METER 0.00062137112
#define _GPS_KM_PER_METER 0.001
//#define _GPS_NO_STATS
class TinyGPS
{
    public:
   TinyGPS();
    bool encode(char c); // process one character received from GPS
   TinyGPS &operator << (char c) {encode(c); return *this;}
   // lat/long in hundred thousandths of a degree and age of fix in milliseconds
   inline void get_position(long *latitude, long *longitude, unsigned long
*fix_age = 0)
   {
     if (latitude) *latitude = _latitude;
```

```
    if (longitude) *longitude = _longitude;
    if (fix_age) *fix_age = _last_position_fix == GPS_INVALID_FIX_TIME ?
      GPS_INVALID_AGE : millis() - _last_position_fix;
  }
  // date as ddmmyy, time as hhmmsscc, and age in milliseconds
  inline void get_datetime(unsigned long *date, unsigned long *time,
unsigned long *fix_age = 0)
  {
    if (date) *date = _date;
    if (time) *time = _time;
    if (fix_age) *fix_age = _last_time_fix == GPS_INVALID_FIX_TIME ?
      GPS_INVALID_AGE : millis() - _last_time_fix;
  }
  // signed altitude in centimeters (from GPGGA sentence)
  inline long altitude() { return _altitude; }

  // course in last full GPRMC sentence in 100th of a degree
  inline unsigned long course() { return _course; }

  // speed in last full GPRMC sentence in 100ths of a knot
  unsigned long speed() { return _speed; }
#ifndef _GPS_NO_STATS
  void stats(unsigned long *chars, unsigned short *good_sentences, unsigned
short *failed_cs);
#endif
   inline void f_get_position(float *latitude, float *longitude, unsigned long
*fix_age = 0)
  {
    long lat, lon;
    get_position(&lat, &lon, fix_age);
    *latitude = lat / 100000.0;
    *longitude = lon / 100000.0;
  }
```

```
  inline void crack_datetime(int *year, byte *month, byte *day,
    byte *hour, byte *minute, byte *second, byte *hundredths = 0, unsigned
long *fix_age = 0)
  {
    unsigned long date, time;
    get_datetime(&date, &time, fix_age);
    if (year)
    {
      *year = date % 100;
      *year += *year > 80 ? 1900 : 2000;
    }
    if (month) *month = (date / 100) % 100;
    if (day) *day = date / 10000;
    if (hour) *hour = time / 1000000;
    if (minute) *minute = (time / 10000) % 100;
    if (second) *second = (time / 100) % 100;
    if (hundredths) *hundredths = time % 100;
  }
  inline float f_altitude()    { return altitude() / 100.0; }
  inline float f_course()      { return course() / 100.0; }
  inline float f_speed_knots() { return speed() / 100.0; }
  inline float f_speed_mph()   { return _GPS_MPH_PER_KNOT *
f_speed_knots(); }
  inline float f_speed_mps()   { return _GPS_MPS_PER_KNOT *
f_speed_knots(); }
  inline float f_speed_kmph()  { return _GPS_KMPH_PER_KNOT *
f_speed_knots(); }
 static int library_version() { return _GPS_VERSION; }
  enum {GPS_INVALID_AGE = 0xFFFFFFFF, GPS_INVALID_ANGLE =
999999999, GPS_INVALID_ALTITUDE = 999999999,
GPS_INVALID_DATE = 0,
   GPS_INVALID_TIME = 0xFFFFFFFF, GPS_INVALID_SPEED =
999999999, GPS_INVALID_FIX_TIME = 0xFFFFFFFF};
  static float distance_between (float lat1, float long1, float lat2, float long2);
```

```
private:
  enum  {_GPS_SENTENCE_GPGGA,  _GPS_SENTENCE_GPRMC,
_GPS_SENTENCE_OTHER};
  // properties
  unsigned long _time, _new_time;
  unsigned long _date, _new_date;
  long _latitude, _new_latitude;
  long _longitude, _new_longitude;
  long _altitude, _new_altitude;
  unsigned long  _speed, _new_speed;
  unsigned long  _course, _new_course;
  unsigned long _last_time_fix, _new_time_fix;
  unsigned long _last_position_fix, _new_position_fix;
  // parsing state variables
  byte _parity;
  bool _is_checksum_term;
  char _term[15];
  byte _sentence_type;
 byte _term_number;
  byte _term_offset;
  bool _gps_data_good;
#ifndef _GPS_NO_STATS
  // statistics
  unsigned long _encoded_characters;
  unsigned short _good_sentences;
  unsigned short _failed_checksum;
  unsigned short _passed_checksum;
#endif
  // internal utilities
  int from_hex(char a);
  unsigned long parse_decimal();
  unsigned long parse_degrees();
  bool term_complete();
```

```
    bool gpsisdigit(char c) { return c >= '0' && c <= '9'; }
    long gpsatol(const char *str);
    int gpsstrcmp(const char *str1, const char *str2);
};
// Arduino 0012 workaround
#undef int
#undef char
#undef long
#undef byte
#undef float
#undef abs
#undef round
#endif
```

Chapter 7: Implementation

Hardware Implementation

The skeletal frame of smart porter is built using wooden planks and wooden sheet. The frame is bonded together with nails and fevicol. Two high torque 12V geared DC motors are controlled via L298 motor driver as wheels for the porter. Arduino powered robot links to the android device via HC-05 Bluetooth module and navigates by comparing its own location by using NEO-6M GPS module to the user's android device location using GPS. the automated smart porter delivers goods within the limited range as it uses a GY-273 HMC5883L compass module and NEO-6M GPS module. The autonomous porter is powered by an Arduino Uno ATmega328p micro-controller. The porter is also provided with a biometric AS608 fingerprint scanner in need of securing your personal possessions. The user will be notified as soon as an individual disconnects from his/her smart porter by an alarming Piezo buzzer.

Software Implementation

The smart porter's is programmed via Arduino ide compiler and it is controlled on an android platformed application built using Android studio. This application provides two functions , one for continuous streaming of smartphones location and one for assigning predetermined GPS way points.

Chapter 8: Testing

Testing of Prototype

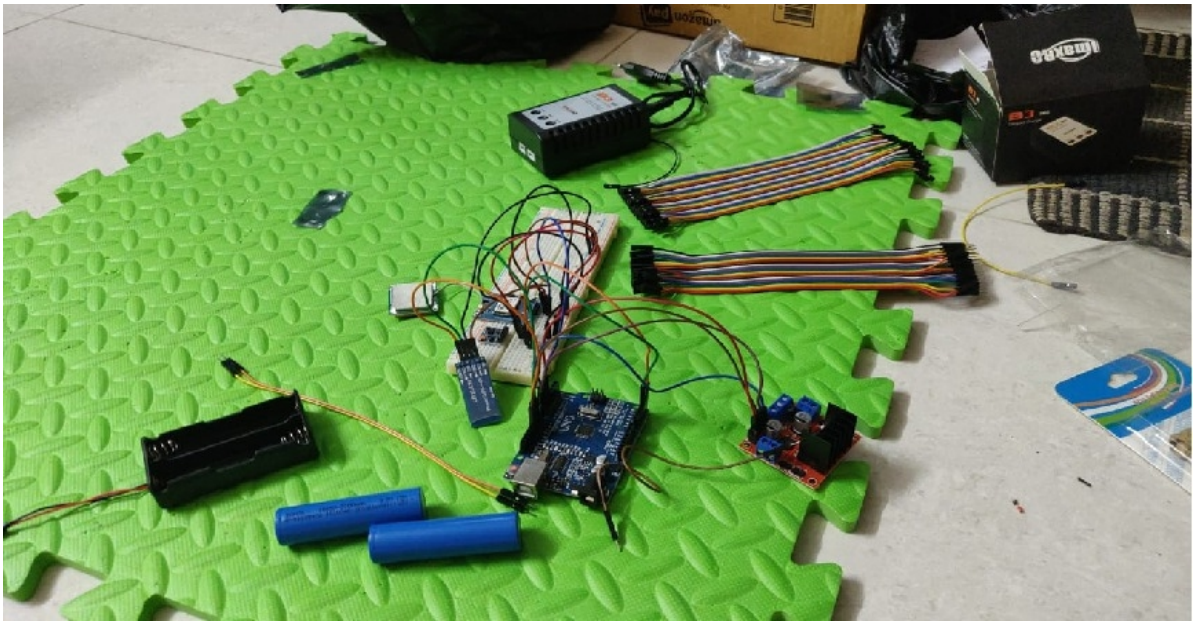


Fig 8.1 Prototype 1

Fig 8.2 Prototype 2

An application was built using Blynk platform in order to operate the smart porter but failed as it couldn't support the Bluetooth functionality.

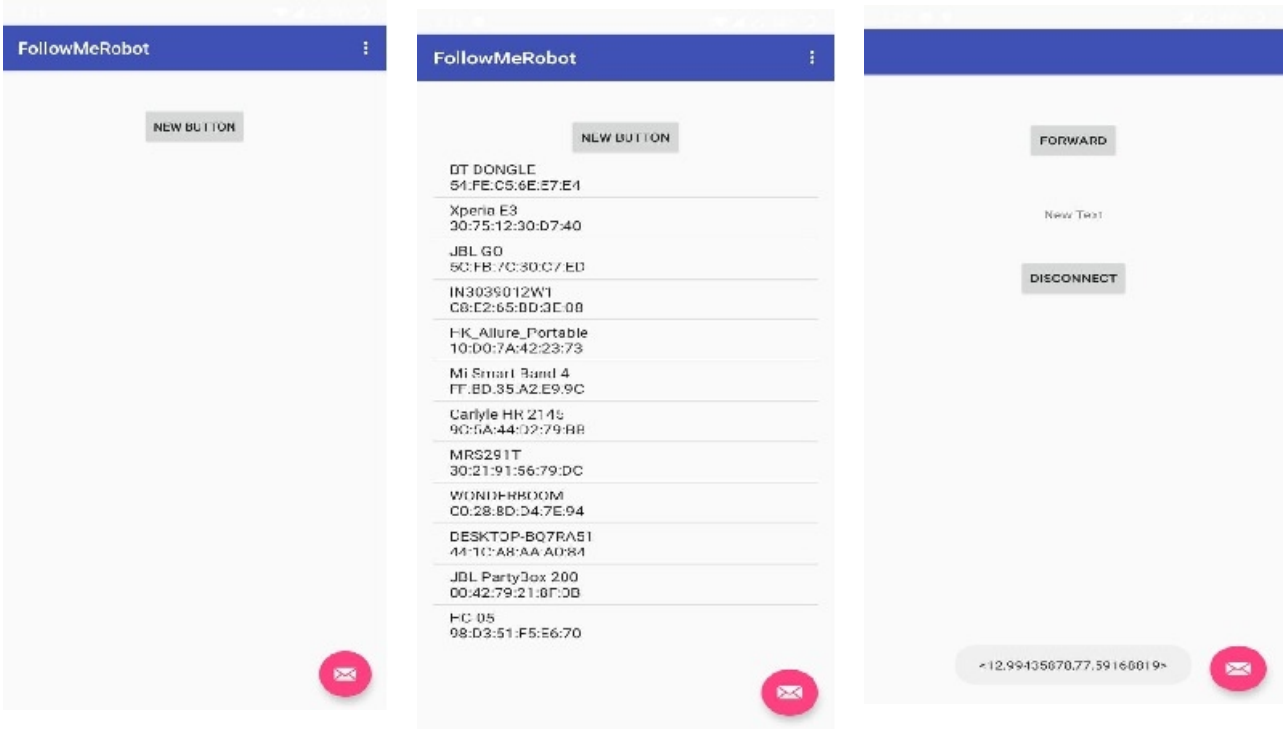


Fig 8.3 android version 1

Fig 8.4 Blynk app

Due to the failure of Blynk application, we built a personalized Android platformed application using Android Studio.

Testing of Real Time Porter

Fig 8.5 Hardware for framework

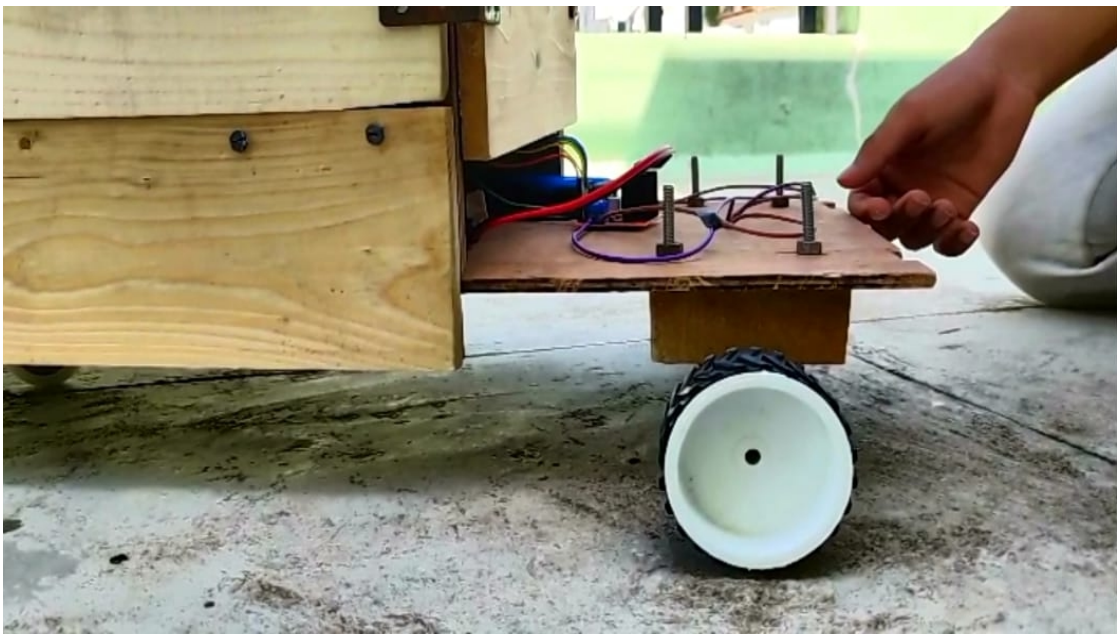


Fig 8.6 Real time porter (slider for easy maintenance)

Fig 8.7 Real time porter

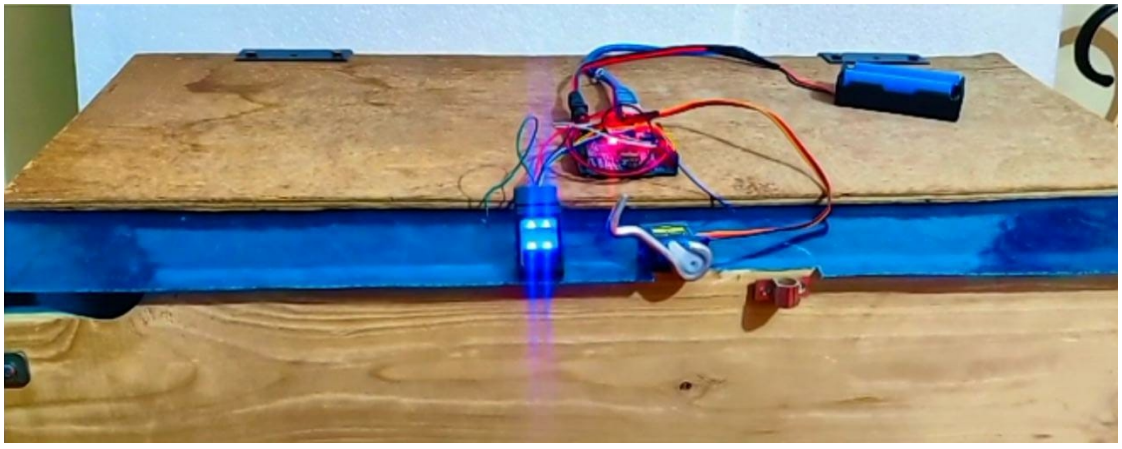


Fig 8.8 Biometric finger scanner lock

Fig 8.9 Real time porter (Volume)

The intelligent porter can port weight up to 16 Kilograms as shown in the figure 8.9.

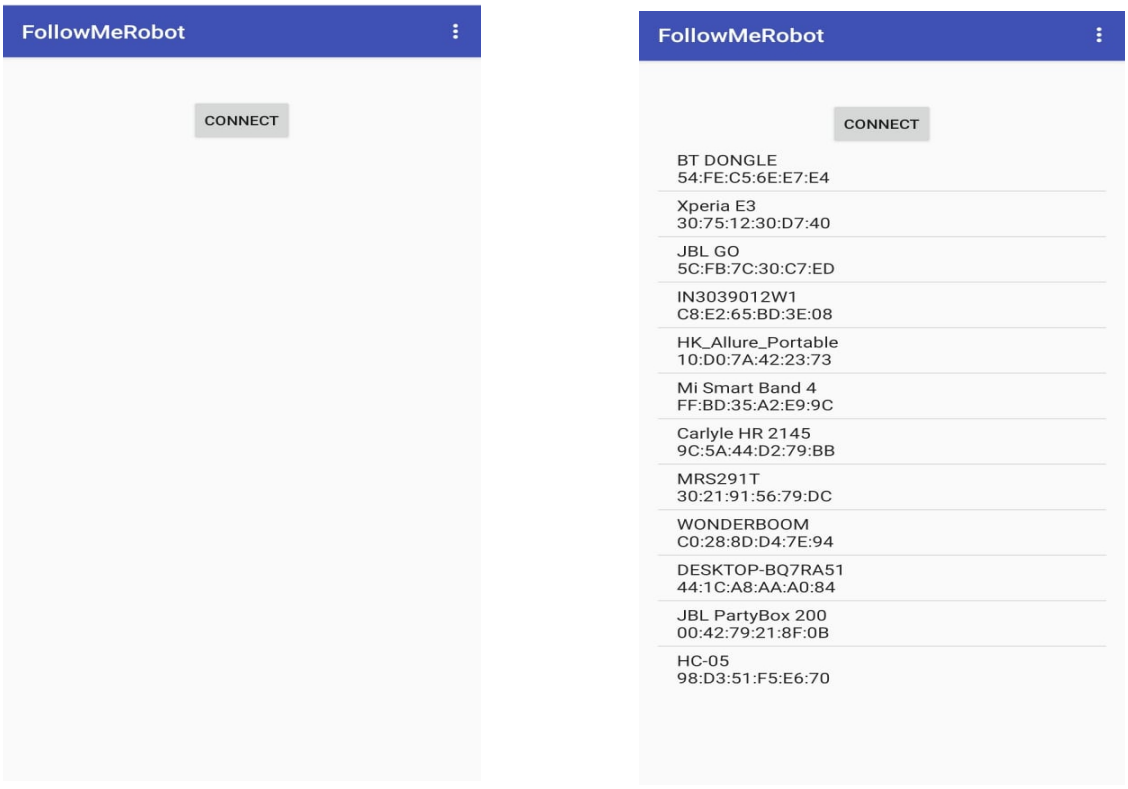


Fig 8.10 Updated android application

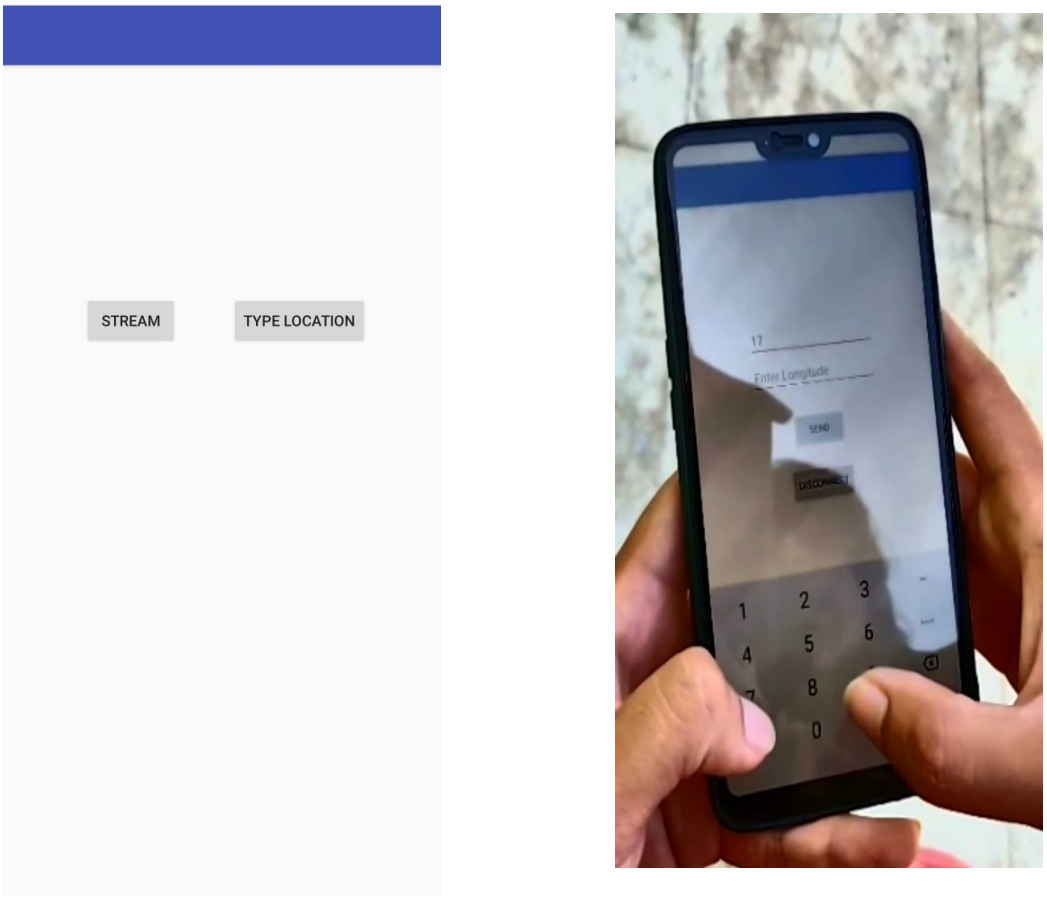


Fig 8.11 Updated android application

The Android application was further updated where one can assign predetermined longitudinal and latitudinal co-ordinates.

Chapter 9: Conclusion

The main objective of Smart porter is to circumvent the challenge of carrying their belongings especially designed for the physically challenged and old age individual. One can avoid spraining and straining themselves which will let them have a hands-free walk . An individual can perform other vital work functions on the move. The porter also safeguards the possessions of the consumer as it provides an advanced security system, a biometric fingerprint sensor lock which is immutable as it can be unlocked only by the user. The porter is equipped with an audio alert device which alarms the consumer when he/she is disconnected from the porter. The porter can be directed via Android application. Reducing manpower and efficiency in terms of reliability, maintainability and future flexibility are its key features.

Appendices

Required

One requires laptop with minimum of 8 GB RAM and 111 GB of SSD. An Android Operating System smart phone or a tablet with good radio frequency (good Bluetooth connectivity). An Ethernet cable to establish connectivity between Arduino Uno and laptop. Male to Female, female to female and male to male jumper wires are required to connect the components with each other. A 12V lippo battery and 7.4V battery are used to power the circuit. A multimeter was used to check the flow of current in the circuit, some of the hardware tools used such as hammer, cutting pliers, wood cutter, jigsaw, nails, soldering gun, insulation tapes, double tapes, clamps, flat wooden file, sandpaper, latches, fevicol, wheels, nuts and bolts.

Not essential but required

One needs good internet connection, earphones to stream project related videos without being disturbed.

References

[1] Bianco R., Caretti M Nolfi S. Developing a robot able to follow a human target in a domestic environment In A. Cesta(Ed.), Proceeding of the First Robocare Workshop. Institute of Cognitive Sciences and Technologies, CNR. Roma: Italy, 2002.

[2] Chuan-Hao Yang "A person-tracking mobile robot using an ultrasonic positioning system" Naval postgraduate school Montery, CA 93943-5000, December 2005.

[3] J. David, N. Cheeke, Fundamentals of ultrasonic waves, Florida, USA: CRC Press, 2002

[4] Keerthi .S. Nair, Anu Babu Joseph, Jinu Isaac Kuruvilla "Design of a low cost human following porter robot at airports" IJACTE, ISSN (Print): 2319-2526, Volume -3, Issue - 2, 2014.

[5]Chuan-Hao Yang" A person-tracking mobile robot using an ultrasonic positioning system" Naval postgraduate school Montery, CA 93943-5000,December 2005.

[6] P. Sai Vamsi, V. Madhava Sarma, S.V.Y.S. Samraj S.R. Deepika, N. Neha, K. Prabhakara Rao undergraduate Student, Professor ECE department, B V Raju Institute of technology, Narsapur, Medak."SMART LUGGAGE" , 2002

[7] Bhanu Prakash Tiwari, Anchal Gupta, Yash Garg, Priyanshu Pandey." Smart Luggage Carrier, 2004

[8] https://github.com/penghez/IoT-Smart-Luggage

[9] https://www.hackster.io/IreneZZhang/smart-suitcase-prototype-13291c

[10] http://www.jurnalteknologi.utm

[11] http:// research.ijcaonline.org

[12] http:// www.researchgate.net

[13]http://www.researchpublish.com/download.php?file=Intelligent