Design & Construction of a Remote Control Forklift Robot

A report submitted to the Department of Mechanical, Sonargaon University of Bangladesh in partial fulfillment of the requirements for the Award of Degree of Bachelor of Science in Mechanical Engineering.

Submitted by

Md Mahfuzur Rahman Mehedi	ID: ME2003022104
Taohidur Rahaman	ID: ME2003022250
Jubayer Al Mahmud Tanjil	ID: ME2003022151
Sale Mahamud Suvo	ID: ME2003022152
Md. Nazmus Sakib	ID: ME2003022128

Supervised by

Md.Ahatashamul Haque Khan Shuvo
Assistant Professor
Department of Mechanical Engineering
Sonargaon University (SU)
Dhaka-1215, Bangladesh

May 15, 2024

Letter of Transmittal

April, 2023

To

Md.Ahatashamul Haque Khan Shuvo

Assistant Professor
Department of Mechanical Engineering.

Sonargaon University (SU)

Subject: Submission of Project Report.

Dear Sir,

We are pleased to submit the project report on **"Design & Construction of a Remote Control Forklift Robot".** It was a great pleasure to work on such an important topic. This project has been done as per instruction of your supervision and according to the requirements of the Sonargaon University.

We expect that the project will be accepted by the concerned authority we will remain happy to further explanation that you may feel necessary in this regard.

Thank You

Sincerely yours,

Md Mahfuzur Rahman Mehedi
ME2003022104

Md. Sale Mahamud Suvo
ME2003022152

Taohidur Rahaman
ME2003022250

Md. Nazmus Sakib
ME2003022128

Md. Jubayer Al Mahmud Tanjil
ME2003022151

DECLARATION

We do hereby solemnly declare that, the work presented here in this project report has been carried out by us and has not been previously submitted to any University/ Organization for award of any degree or certificate

We hereby ensure that the works that has been prevented here does not breach any existing copyright.

We further undertake to indemnify the university against any loss or damage arising from breach of the foregoing obligation.

Md Mahfuzur Rahman Mehedi
ME2003022104

Md. Sale mahamud suvo
ME2003022152

Taohidur Rahaman
ME2003022250

Md. Nazmus Sakib
ME2003022128

Md. Jubayer Al Mahmud Tanjil
ME2003022151

ACKNOWLEDGEMENT

First, we started in the name of almighty Allah. This thesis is accomplished under the supervision of **Md.Ahatashamul Haque Khan shuvo**, Assistant Professor, Department of Mechanical, Sonargaon University. It is a great pleasure to acknowledge our profound gratitude and respect to our supervisor for this consistent guidance, encouragement, helpful suggestion, constructive criticism and endless patience through the progress of this work. The successful completion of this thesis would not have been possible without his persistent motivation and continuous guidance.

The author are also grateful to Md. Mostofa Hossain, Head of the Department of Mechanical Engineering and all respect teachers of the Mechanical Engineering Department for their co-operation and significant help for completing the thesis work successfully.

[Authors]

Md Mahfuzur Rahman Mehedi
ME2003022104

Md. Sale mahamud suvo
ME2003022152

Taohidur Rahaman
ME2003022250

Md. Nazmus Sakib
ME2003022128

Md. Jubayer Al Mahmud Tanjil
ME2003022151

Abstract

Forklifts are used in a wide variety of applications, such as manufacturing, construction, retail, meat and poultry processing, lumber and building supplies, trades, agriculture, and a variety of warehouse operations. This work mainly approaches to design and modeling of electronic forklift. Electronic forklifts are extensively used primarily for material handling in food industry. The objective of the electronic forklift is to ensure safety of the operator and to save time and money to push and pop items. In this system, 4 DC motors, 2 gear moto, A Node MCU, Motor Driver, r are used. Four DC motors are used for moving and two gear motor is used for lifting and Node MCU is mainly used to control the overall system. Node MCU will determine whether the motors have to rotate forward or backward. Motor directions are implemented by Arduino programming management. Therefore, the system will be a foundation in implementing of the industrial forklift.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Introduction

The forklift can be defined as a device capable of lifting heavy loads. A forklift is a vehicle similar to a small truck that has two metal forks on the front used to lift load. The forklift operator drives the forklift forward until the forks push under the load. Since it operates through a remote, it doesn't contain any steering mechanism. The entire vehicle is designed to drive through four wheels &motors are used to drive all the four wheels directly. All the motors are driven through a single „H" bridge DC motor drive package. The DC Motors are having reduction gear mechanism internally, there by speed is reduced and torque is increased. There is also a lead screw mechanism which is used for linear mechanism of fort for load lifting work which is driven by side shaft motor. The mainly is about designing and fabricating forklift which is operated by wireless controlled. The lift and movement system of the forklift will be provided by miniature electric-powered motor. This forklift can move

Forward backward, turning right and left depend on the signal from wireless module. The objective of this project is to design the forklift which includes the physical structure, electronic circuit, control and operating system. The research on this field has great functions and benefits and can be proceed in the future. Maneuverable and require only one operator to lift, transport, and stack or unstack the material.

Forklifts may be used for indoor or outdoor use depending on their size, tires and load capacities. For warehouse use because they do not give off noxious fumes like gas powered machines do. Forklifts are most often used in warehouses, but some are meant to be used outdoors. The vast majority of rough terrain forklifts operate on gasoline, but some use diesel or natural gas. Rough terrain forklifts have the highest lifting capacity of all forklifts and heavy duty tires (like those found on trucks), making it possible to drive them on uneven suWI-FI aces outdoors. It is important for forklift operators to follow all safety

precautions when using a forklift. Drivers should be careful not to exceed the forklift's weight capacity. Forklift operators also need to be able to handle forklift's rear wheel steering. Driving a forklift is similar to driving a car in reverse, meaning that the driver must constantly steer to keep it moving in a straight line. The driver must be aware of the forklift's ever-changing center of gravity and avoid making any quick sharp turns or going too fast. It is advisable that anyone who operates a forklift be fully trained and licensed.

1.2 Objective:

The objectives of this project are:

- To design and construct a **Remote Control Forklift Robot**
- To test the performance of the Remote Controlled Mini Forklift Robot.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review

The forklifts are the essential part of today's supply unit where it is transporting goods from one place to one more in the storage, load, or unloading good to the truck the forklift utilizes. The two-wheel hand over operate forklift first appear over one hundred years ago. These original units were shaped-iron axle and cast-iron wheels which enabled many to be lifted and transported without physical labor. The evolution to unite horizontal and vertical motion resulted in the first hand operate forklift capable of exciting a few inches of the land and with this the growth of the forklift is on Fast Track till today. In 1917 the business named Clark started the manufacturer of axles and created a truck called the tractor to move resources around their factory in a variety of units of the plants. As visitors to the industrial unit saw the tractor at work, they are also paying attention in the way to use a tractor in their place live, so they placed orders to build tractor for their business. Both of these developments increased making of forklifts and allowed distributors the means to professionally shift heavy many in the factories.

'Load centers represent the center of a forklift truckload, from front to rear. It is easily calculated by measuring the load to be carried and in-between by two (providing the load is evenly distributed and located to butt up to the forklift backrest). The joint center of the seriousness of the truck and weight system shifts onward outside the constant triangle, as the load's moment, is greater than the vehicle's instant, and the forklift tips forward, Pivoting on the front axle or fulcrum. The regular forklift is wide in size and cannot be utilized in a narrow passage with the load as there are having more chase of an accident as it's fairly difficult to rotate n control in narrow passage whereas the two-wheeler forklifts can be utilized very easily narrow passages as compared to a normal forklift.

2.2 Summary

From the literature discussed above, we gained a lot of knowledge and we inspired to do this project. We were able to do it with everyone tireless work.

CHAPTER 3

THEORITICAL MODEL

3.1 Introduction

In this chapter we will discuss about our system methodology, system description, block diagram, circuit diagram, working principle, instrument cost analysis.

3.2 Methodology

The Methodology of this project are:

- Creating an idea for design and construction of **"Remote Control Forklift Robot".** And designing a block diagram & circuit diagram to know which components we need to construct it.

- Collecting all the components and programming the micro-controller to control the system.
- Setting up all the components in a PCB board & then soldering. Then assembling all the blocks in a board and finally running the system & checking.

3.3 Block Diagram:

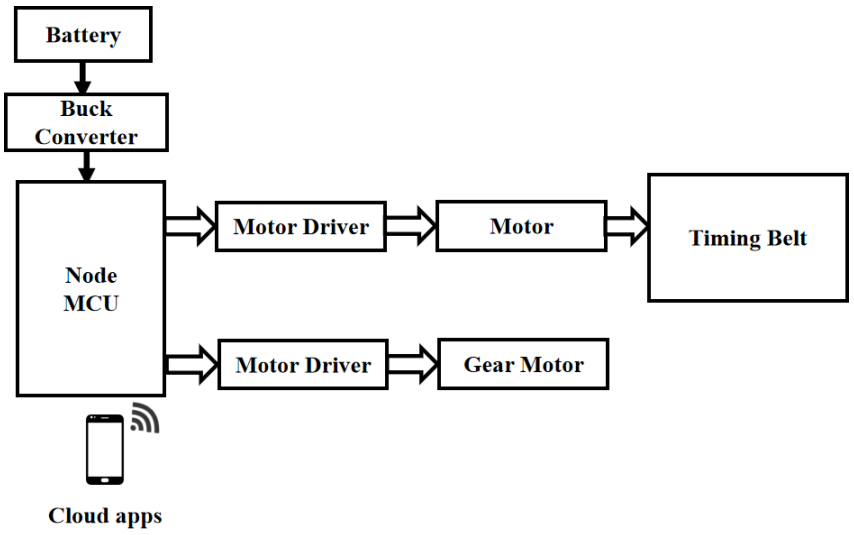

Figure 3.1: Block Diagram of A Remote Control Forklift Robot.

3.4 Circuit Diagram

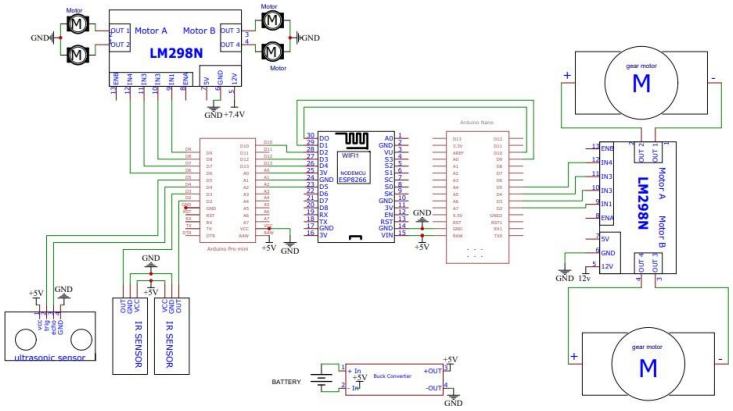

Figure 3.2: Schematic Diagram of A Remote Control Forklift Robot.

3.5 Complete Project Image

Figure 3.3: Our Final Project Prototype

3.6 Working Principle

Here we use a Node MCU micro-controller which is the main controller unit for this system. This micro-controller is mainly powered from the battery. The working principle of the forklift is based on the micro-controller receiving signals from the remote control and sending signals to the motor drivers, which then control the motors. The micro-controller is programmed to interpret the signals from the remote control and convert them into specific actions, such as moving the forklift forward, backward, left, or right, or raising and lowering the forks. The motor drivers regulate the power supplied to the motors, controlling their speed and direction of rotation. As the motors turn, they rotate the wheels and control the movement of the forklift. The forklift can be powered by a rechargeable battery, which provides the necessary electrical energy to the system. Overall, the working principle of a remote controlled mini forklift involves the integration of various components that work together to control the movement of the vehicle, enabling it to move, lift, and transport objects with ease. This is the main procedure of this system.

3.7 Components List:

Hardware Part:

1. Node MCU
2. IR Sensor
3. Battery
4. Gear Motor
5. Motor Driver

Software Part:

1. Arduino IDE
2. EasyEDA
3. Blynk Apps

3.9 Node MCU

Node MCU is an open-source firmware for which open-source prototyping board designs are available. The name "Node MCU" combines "node" and "MCU" (micro-controller unit). The term "Node MCU" strictly speaking refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are open source. The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

Figure 3.7: Node MCU

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

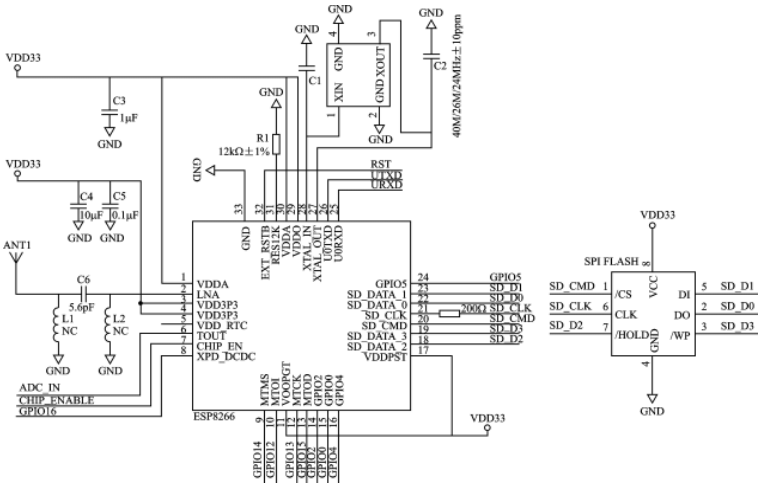

Figure 3.8: Node MCU Schematic Diagram

This an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Express if Systems, and hardware which is based on the ESP-12 module. The term "Node MCU" by default refers to the firmware rather than the development kits. The firmware uses the Luascripting language. It is based on the eLua project, and built on the Espress if Non-OS SDK for ESP8266. Node MCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems [6] began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). Node MCU started on 13 Oct 2014, when Hong committed the first file of node mcu-firmware to GitHub.[11] Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9.[12] Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform,[13] and committed to NodeMCU project, then Node MCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glibto Node MCU project,[15] enabling Node MCU

to easily drive LCD, Screen, OLED, even VGA displays. In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over.

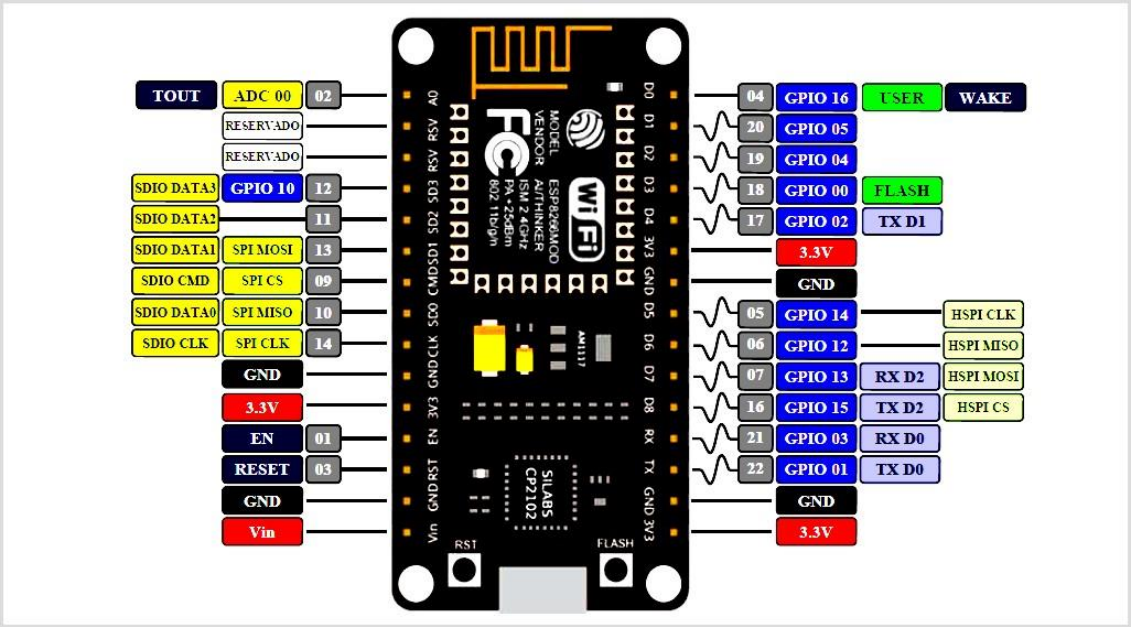

Figure 3.9: Node MCU Pin Out

Node MCU V3 ESP8266 ESP-12E is Wi-Fi development board that helps you to prototype your IoT product with few Lua script lines, or through Arduino IDE. The board is based on ESP8266 ESP-12E variant, unlike other ESP-12E, you won't need to buy a separate breakout board, USB to serial adapter, or even solder it to a PCB to get started, you will only need a USB cable (Micro USB).

Features

1. Communication interface voltage: 3.3V.
2. Antenna type: Built-in PCB antenna is available.
3. Wireless 802.11 b/g/n standard
4. WiFi at 2.4GHz, support WPA / WPA2 security mode

5. Support STA/AP/STA + AP three operating modes
6. Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)
7. D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA
8. AD0: 1 channel ADC
9. Power input: 4.5V ~ 9V (10VMAX), USB-powered
10. Current: continuous transmission: ≈70mA (200mA MAX), Standby: <200uA
11. Transfer rate: 110-460800bps
12. Support UART / GPIO data communication interface
13. Remote firmware upgrade (OTA)
14. Flash size: 4MByte.

3.12 Motor driver

The **L293D** is a popular 16-Pin **Motor Driver** IC. As the name suggests it is mainly used to drive **motors**. A single **L293D** IC is capable of running two DC **motors** at the same time; also the direction of these two **motors** can be controlled independently.

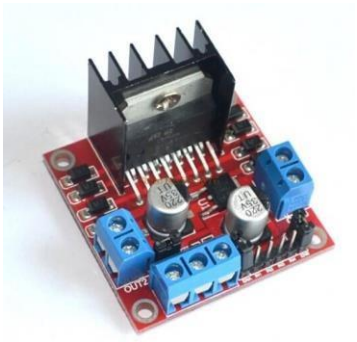

Figure 3.14: Motor driver IC L293D

Working Process:

L293D IC is a typical **Motor Driver** IC which allows the DC **motor** to drive on any direction. This IC consists of 16-pins which are used to control a set of two DC **motors** instantaneously in any direction. It means, by using a **L293D** IC we can control two DC **motors**.

Features

- Can be used to run Two DC motors with the same IC.
- Speed and Direction control is possible
- Motor voltage Vcc2 (Vs): 4.5V to 36V
- Maximum Peak motor current: 1.2A
- Maximum Continuous Motor Current: 600mA
- Supply Voltage to Vcc1(vss): 4.5V to 7V
- Transition time: 300ns (at 5Vand 24V)
- Automatic Thermal shutdown is available

L293D Pin Configuration

Pin Number	Pin Name	Description
1	Enable 1,2	This pin enables the input pin Input 1(2) and Input 2(7)
2	Input 1	Directly controls the Output 1 pin. Controlled by digital circuits
3	Output 1	Connected to one end of Motor 1
4	Ground	Ground pins are connected to ground of circuit (0V)
5	Ground	Ground pins are connected to ground of circuit (0V)
6	Output 2	Connected to another end of Motor 1

7	Input 2	Directly controls the Output 2 pin. Controlled by digital circuits
8	Vcc2 (Vs)	Connected to Voltage pin for running motors (4.5V to 36V)
9	Enable 3,4	This pin enables the input pin Input 3(10) and Input 4(15)
10	Input 3	Directly controls the Output 3 pin. Controlled by digital circuits
11	Output 3	Connected to one end of Motor 2
12	Ground	Ground pins are connected to ground of circuit (0V)
13	Ground	Ground pins are connected to ground of circuit (0V)
14	Output 4	Connected to another end of Motor 2
15	Input 4	Directly controls the Output 4 pin. Controlled by digital circuits
16	Vcc2 (Vss)	Connected to +5V to enable IC function

Use of a L293D Motor Driver IC:

Using this L293D motor driver IC is very simple. The IC works on the principle of **Half H-Bridge**, let us not go too deep into what H-Bridge means, but for now just know that H bridge is a setup which is used to run motors both in clock wise and anticlockwise direction. As said earlier this IC is capable of running two motors at the any direction at the same time, the circuit to achieve the same is shown below.

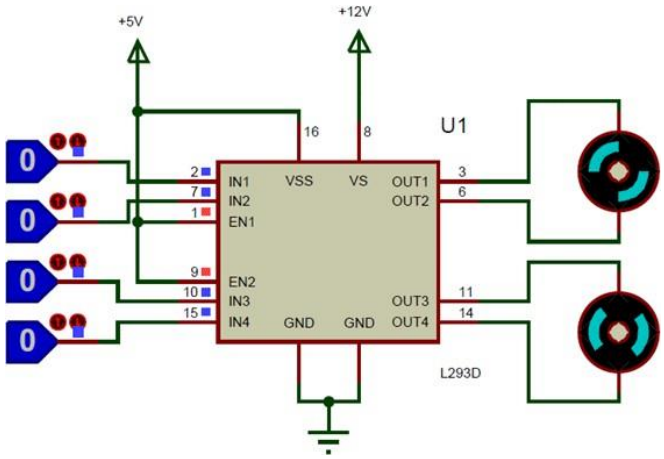

Figure 3.15: L293D circuit Diagram

All the Ground pins should be grounded. There are two power pins for this IC, one is the Vss(Vcc1) which provides the voltage for the IC to work, this must be connected to +5V. The other is Vs(Vcc2) which provides voltage for the motors to run, based on the specification of your motor you can connect this pin to anywhere between 4.5V to 36V, here I have connected to +12V.

The Enable pins (Enable 1,2 and Enable 3,4) are used to Enable Input pins for Motor 1 and Motor 2 respectively. Since in most cases we will be using both the motors both the pins are held high by default by connecting to +5V supply. The input pins Input 1,2 are used to control the motor 1 and Input pins 3,4 are used to control the Motor 2. The input pins are connected to the any Digital circuit or micro controller to control the speed and direction of the motor.

Applications

- Used to drive high current Motors using Digital Circuits
- Can be used to drive Stepper motors
- High current LED’s can be driven

3.13 DC Gear Motor

Electric motors turn electricity into motion by exploiting electromagnetic induction. A simple direct current (DC) motor is illustrated below. The motor features a permanent horseshoe magnet (called the stator because it's fixed in place) and an turning coil of wire called an armature (or rotor, because it rotates).

Figure 3.16: DC Gear motor

Motor Specifications

- Standard 130 Type DC motor
- Operating Voltage: 4.5V to 9V
- Recommended/Rated Voltage: 6V
- Current at No load: 70mA (max)
- No-load Speed: 9000 rpm
- Loaded current: 250mA (approx.)
- Rated Load: 10g*cm
- Motor Size: 27.5mm x 20mm x 15mm

Pin Description

Table 01: DC Gear Motor Pin Out

No:	Pin Name	Description
1	Terminal 1	A normal DC motor would have only two terminals. Since these terminals are connected together only through a coil they have not polarity. Revering the connection will only reverse the direction of the motor
2	Terminal 2	

Use of the DC motor:

As the name suggests the Hobby DC motor is highly used by hobbyists who start exploring electronics. Hence this motor is very simple and easy to use. You can use any normal 9V battery or even a 5V supply since this motor has a operating range from 4.5V to 9V. In order to make it rotate just connect the positive (+) side of battery to one terminal and the Negative (-) sign of the battery to the other end and you should see the motor rotating. If you want to reverse the speed of the motor simply interchange the terminals and direction will also be reversed. In order to control the speed of the motor you have to vary the voltage supplied to the Motor the easiest way to do this is using a Potentiometer. There are also many other way to achieve this. Also remember that the motor can consume upto 250mA during loaded conditions so make sure you supply could source it. If you are controlling it through any Digital IC or any Micro-controller you should use a motor driver IC like L293D or ULN2003

Applications:

- Toy cars
- Windmill projects
- Basic Electronics projects
- As Robot wheels

3.14 5V Regulator IC

Voltage sources in a circuit may have fluctuations resulting in not providing fixed voltage outputs. A voltage regulator IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series of fixed linear voltage regulators used to maintain such fluctuations, is a popular voltage regulator integrated circuit (IC). The xx in 78xx indicates the output voltage it provides. 7805 IC provides +5 volts regulated power supply with provisions to add a heat sink.

7805 IC Rating:

- Input voltage range 7V- 35V
- Current rating Ic = 1A
- Output voltage range V. Max=5.2V ,V. Min=4.8V

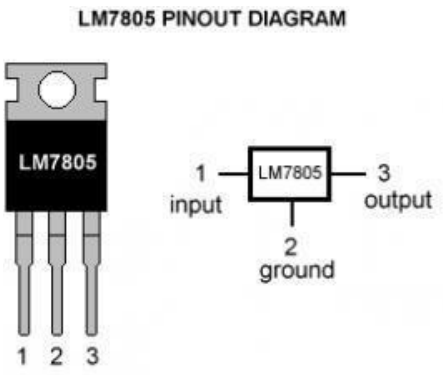

Figure 3.17: 5V Regulator IC

3.15 Battery

Lithium batteries are primary batteries that have metallic lithium as an anode. These types of batteries are also referred to as lithium-metal batteries. They stand apart from other batteries in their high charge density and high cost per unit. Depending on the design and chemical compounds used, lithium cells can produce voltages from 1.5 V (comparable to a zinc–carbon or alkaline battery) to about 3.7 V.

Disposable primary lithium batteries must be distinguished from secondary lithium-ion or a lithium-polymer,[1] which are rechargeable batteries. Lithium is especially useful, because its ions can be arranged to move between the anode and the cathode, using an intercalated lithium compound as the cathode material but without using lithium metal as the anode material. Pure lithium will instantly react with water, or even moisture in the air; the lithium in lithium ion batteries is in a less reactive compound.

Lithium batteries are widely used in portable consumer electronic devices. The term "lithium battery" refers to a family of different lithium-metal chemistries, comprising many types of cathodes and electrolytes but all with metallic lithium as the anode. The battery requires from 0.15 to 0.3 kg of lithium per kWh. As designed these primary systems use a charged cathode, that being an electro-active material with crystallographic vacancies that are filled gradually during discharge.

Figure 3.18: 3.7V Battery

Product Specification

Voltage	3.7 V
Product Type	Lithium-Ion
Battery Capacity	2200mAh
Weight	45 g

3.17 Arduino IDE

The digital micro-controller unit named as Arduino Nano can be programmed with the Arduino software IDE. There is no any requirement for installing other software rather than Arduino. Firstly, Select "Arduino Nano from the Tools, Board menu (according to the micro-controller on our board). The IC used named as ATmega328 on the Arduino Nano comes pre burned with a boot loader that allows us to upload new code to it without the use of an external hardware programmer.

Communication is using the original STK500 protocol (reference, C header files). We can also bypass the boot loader and programs the micro-controller through the ICSP (In Circuit Serial Programming) header. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

The Arduino Nano is one of the latest digital micro-controller units and has a number of facilities for communicating with a computer, another Arduino, or other micro-controllers. The ATmega328 provides UART TTL at (5V) with serial communication, which is available on digital pins 0 -(RX) for receive the data and pin no.1 (TX) for transmit the data. An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an .in file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board.

The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial Communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Nano's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. Arduino programs are written in C or C++ and the program code written for Arduino is called sketch. The Arduino IDE uses the GNU tool chain and AVR Lab to compile programs, and for uploading the programs it uses argued. As the Arduino platform uses Atmel micro-controllers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

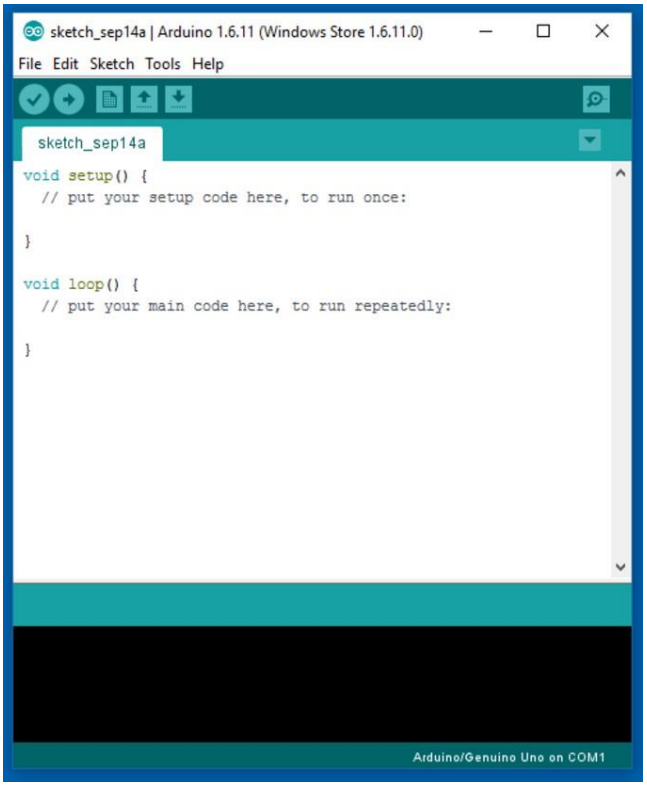

Figure 3.20: Arduino Software Interface IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a

text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.hBefore uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for a Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload.

On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino Boot loader, a small program that has been loaded on to the micro-controller on your board. It allows you to upload code without using any additional hardware. The Boot loader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the micro-controller. The Boot loader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, Boot loaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory.

Don't use "Arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor.

3.18 EasyEDA Software

EasyEDA is a web-based EDA tool suite that enables hardware engineers to design, simulate, share - publicly and privately - and discuss schematics, simulations and printed circuit boards. Other features include the creation of a bill of materials, Gerber files and pick and place files and documentary outputs in PDF, PNG and SVG formats.EasyEDA allows the creation and editing of schematic diagrams, SPICE simulation of mixed analogue and digital circuits and the creation and editing of printed circuit board layouts and, optionally, the manufacture of printed circuit boards.

Subscription-free membership is offered for public plus a limited number of private projects. The number of private projects can be increased by contributing high quality public projects, schematic symbols, and PCB footprints and/or by paying a monthly

subscription.Registered users can download Gerber files from the tool free of charge; but for a fee, EasyEDA offers a PCB fabrication service. This service is also able to accept Gerber file inputs from third party tools.

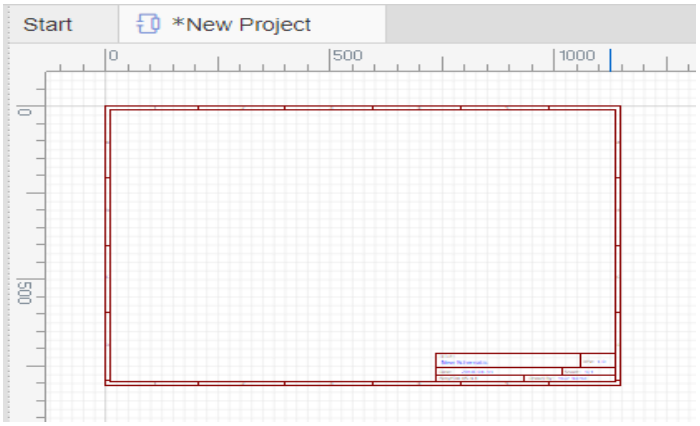

Figure 3.21: EasyEDA Software Interface

3.19 Blynk Apps

Blynk is an **Internet-of-Things** platform designed to make development and implementation of smart IoT devices quick and easy. It can be used to read, store, and visualize sensor data and control hardware remotely.

Figure 3.22: Blynk Apps

Internet of Things has been all the buzz lately and more and more devices are being talking to internet every day. With the rise of such amazing technology, the risk of security has also increased substantially. Some of the major concerns in IoT are:

- If IoT devices are sending your data to the internet, the communication needs to be closed and encrypted which cannot be possible without using a dedicated and closed server which is really hard to manage.
- The IoT devices also need to be responsive and again, that is not possible without a server with low latency and high responsiveness.
- In IoT, the platform needs to be compatible with many different types of hardware architecture and devices, so that it doesn't restrict its users with single type of hardware with limited capabilities.

Keeping in view the problems mentioned above, Blynk is the perfect solution for all these problems. Blynk consists of the following three major components:

Blynk App – The mobile app developed by Blynk works as a control panel for visualizing and controlling your hardware. It is available for both Android and iOS. The app offers a very productive interface and various different widgets for different purposes. Blynk works on a currency of its own called energy. New users get 2000 amount of Blynk energy with a free Blynk account and this energy is used to buy and deploy widgets in the projects.

Blynk Server – The most amazing component of the Blynk Platform which makes it all possible is the Blynk Server. Blynk offers a secure, responsive and centralized cloud service through its server which allows all of this communication between the devices. The Blynk server is also available as open source so you can literally make your own server and make it even more secure with a little tinkering.

Blynk Library – The key feature of Blynk platform which makes it scalable and amazing, is the Blynk Library. The Blynk Library makes it possible to connect your hardware and get it up and running in a blink. The support for multiple hardware devices including **Arduino, ESP8266** and **Raspberry Pi** is included in the library and it also makes it possible to connect with hardware through many different ways of communication like Wi-Fi, Bluetooth, BLE, USB and GSM.

Features:

- Similar API & UI for all supported hardware & devices
- Connection to the cloud using:
    - Wi-Fi
    - Bluetooth and BLE
    - Ethernet
    - USB (Serial)
    - GSM
    - Set of easy-to-use Widgets
- Direct pin manipulation with no code writing
- Easy to integrate and add new functionality using virtual pins
- History data monitoring via Super Chart widget
- Device-to-Device communication using Bridge Widget
- Sending emails, tweets, push notifications, etc

CHAPTER 4
Results & Discussion

4.1 Result Analysis

Now, it's time to talk about the results. We have written our commands using the Arduino IDE and the following things can happen: Our project is **Design & Construction of a Remote Control Forklift**. In our project making we used PVC boards for total hardware making. After finally completing this project, we run it & we observed the output of this project. We can see that it is working well as expected.

- After power on our project will be able to work.
- Then the working result of a Line Flowing and Remote Control Lifting Mini Forklift is a small, battery-powered vehicle that can be controlled remotely using a handheld controller.
- Then it typically includes a chassis with wheels, motors, and a battery, as well as a micro-controller that controls the motors and responds to the remote control commands.
- Presses the buttons on the Blynk Apps, the micro-controller sends signals to the motor drivers, which in turn control the motors. This allows the mini forklift to move forwards and backwards, turn left and right, and raise and lower its forks.
- All Control and show this project of Blynk App.

Example:

Design of load

p = pitch of the load = 1.5mm.

d = diameter of the load = 10mm.

m = mass carried by the load = 1.5kg.

g = Acceleration due to gravity = 9.81ms-2

Total load –

1. Total Load (W) = Mass (m) × Acceleration due to gravity (g) = 1.5*9.81 = 14.715N.

2. Mean Diameter of the load (dm) –

Mean diameter (dm) = d-(0.5*p) = 10-(0.5*1.5) = 4.25.25mm.

3 Helix Angle of the lead screw (α) –

The lead screw has single start threads, Therefore l= p = 1.5 mm

$\tan\alpha = l/(\pi * dm) = 1.5/(\pi * 4.25)$ $\alpha = \tan^{-1}(0.031644)$ $\alpha = 2.11°$

Coefficient of friction(μ) = 0.08 tan

$\emptyset = \mu$ $\emptyset = \tan^{-1}(0.08)$ $\emptyset = 4.57$

Torque required to lift and lower the load –

Torque For lifting:

T1 = Torque required for lifting the load

$T1 = (W*dm)/2 \times \tan(\emptyset+\alpha) = 14.715*4.25/2 * \tan(2.11 + 4.57) = 31.26*0.11 = 3.43$ N-

For lowering:

T2 = Torque required for lowering the load

$T2 = (W*dm)/2 \times \tan(\emptyset-\alpha) = 14.25*4.25/2 * \tan(4.57 - 2.95) = 31.26*0.04 = 1.34$ N-mm.

Holding Torque of the Motor (Th) –

Voltage of the motor (V) = 6v

Current of the motor (I) = 2 amp

Speed of the motor = 60rpm

Power (P) = V*I =6*2 =12watt

Holding torque (Th) = $p*60/2\pi$

= $12*60/2*\pi*60$ = 1.90 N-m = 19 kg-cm

Mechanical Efficiency (ɳ)-

ɳ = $tan\alpha$/tan(α+Ø) = tan(2.11)/tan(2.11+4.57) = 0.3145 = 0.3145*100 = 31.45%

Result Analysis Table:

Load	Helix angle of load	Friction Angle	Torque for Lifting	Torque for lowering	Holding Torque	Efficiency
1kg	2.11°	4.57	2.29N-mm	0.85N-mm	19kg-cm	31.45
1.5kg	2.11°	4.57	3.43N-mm	1.34N-mm	19kg-cm	31.45
2kg	2.11°	4.57	4.58N-mm	1.66N-mm	19kg-cm	31.45
5kg	4.046°	4.57	33.44N-mm	2.018N-mm	19kg-cm	46.68

4.2 Advantages

There are certainly many advantages of our project and some of the major ones have been given below:

- Increased safety
- Improved Efficiency
- Versatility.

4.3 Application

This project has applications in many fields due its necessity. We have selected a few of them and they are given below:

- It can be used in any Warehouses.
- It can be used in any industry or corporate offices.
- It can be used in manufacturing.
- It can be used in Agriculture.

4.4 Discussion

The safety of the operator can be improved by providing alternatives to the existing design of the stand up forklift trucks. One such alternative is to provide a door to protect the operator. Design changes had to be made to prevent accidents and to protect the operator in case of an accident. In the electronic forklift, by using the electronic control system, the operator doesn't face any difficulties in the work envelope. Regular preventative maintenance is required.

CHAPTER 5
CONCLUSION

5.1 Conclusion

The technology is to add to the safety of the operator by operating the forklift from a sure distance. This increases the efficiency of the output because human error due to poor visibility can be minimized. The system is designed and developed productively, for the display purpose prototype model is constructed. Most of all human safety is a major concern by using the remotely operated forklift. Our project has an easy electrical heart and a simple mechanical body. As this is the simplest one, we have got a wired remote for manual working it can be modified into any high-class use. Considering the project time and all the necessary steps, we concluded this project is the true one. Since just a simple change in its mechanical arm and movement way, we can convert it into any robot that can perform a special type of work. It can be modified into any high-class function we concluded that remote prohibited forklift is the only way to stop such industrial issues like labor cost, dangerous material handling.

5.2 Future Work

As we have already discussed about the advantages, applications of our project so definitely there's room for improvement. Some of these are listed below:

- In future, we are thinking about adding sensors.
- In future, we are thinking about adding automation.
- In future development can be Improved Battery technology.
- In future, we are thinking about adding camera monitoring for security.

Reference

[1] Dr.R.N. Mall, Automated Guided Vehicle, ISBN 2091 Journal, MMMEC, Gorakhpur. (2013)

[2] Kenneth B. Ackerman, Forklifts and Other Mobile Equipment, Practical Handbook of Warehousing. (1990)

[3] RS Khurmi, J.K Gupta, A textbook of Machine Design. (2005)

[4] SS Rattan, Theory of Machines, Professor of Mechanical Engineering, National Institute of Technology, Kurukshetra. (2009)

[5] VB Bhandari (Design of Machine Elements, Retired Professor and Head Department of Mechanical Engineering, Vishwakarma Institute of Technology, Pune. (2010)

[6] JB Gupta,Basic Electrical & Electronics Engineering.(2011)

[7] B L Thareja, A K Thareja Revised by S G Tarnekar, Electrical Technology, Former Professor & Head, Electrical Engineering Department, Visvesaraya National Institute of Technology, Nagpur. (2005)

[8] From Vol. IV Number 1 of Warehousing Forum, The Ackerman Co (1988)

[9] Conte M, Pasquali M, Impact of innovative ILHYPOS super capacitors on a fuel cell vehicle, international electric vehicle symposium EVS-24. Stavanger, Norway (2009)

[10] Contem, Super capacitors technical requirements for new applications. Fuel Cells10:806–8(2010)

[11] Jagalamudi, "Failure Rate Studies and Design Alternatives for Standup Forklift Trucks", September 2004.

[12] Thomas E.Kissell, Industrial Electronics, Prentice-Hall International, Inc.

[13] L.Floyd Thomas, Electronics Devices, Prentice-Hall international, Inc.

[14] Webb Greshok, Industrial Control Electronics, Macmillan.

[15] 1998. David J.Comer,Donald T.Comer, Advanced Electronics, Second Edition,published by Paperback, McGraw-Hall.

[16] Valentine R: Motor Control Electronic Handboo, McGraw - Hill, NewYork (1998).

Appendix

Programming Code:

```
#define BLYNK_TEMPLATE_ID "TMPLZE8eZcL8"
#define BLYNK_TEMPLATE_NAME "zeronebd  2"
#define BLYNK_AUTH_TOKEN "Y3O17_RpEGvzgUPqdDwvlpDDZhpmC5cp"
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial



#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "abcde";
char pass[] = "123456789";

int p0=D0;
int p1=D1;
int p2=D2;
int p3=D3;
int p4=D4;
int p5=D5;
int p6=D6;
int p7=D7;
```

```
// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin V1
BLYNK_WRITE(V0)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p0,pinValue);
  delay(100);
  // process received value
}
BLYNK_WRITE(V1)
{
  int pinValue1 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p1,pinValue1);
  delay(100);
  // process received value
}
BLYNK_WRITE(V2)
{
  int pinValue2 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p2,pinValue2);
  delay(100);;
  // process received value
}
BLYNK_WRITE(V3)
{
  int pinValue3 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p3,pinValue3);
  delay(100);
  // process received value
```

```
}
BLYNK_WRITE(V4)
{
  int pinValue4 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p4,pinValue4);
  delay(100);
  // process received value
}
BLYNK_WRITE(V5)
{
  int pinValue5 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p5,pinValue5);
  delay(100);
  // process received value
}
BLYNK_WRITE(V6)
{
  int pinValue6 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p6,pinValue6);
  delay(100);
  // process received value
}
BLYNK_WRITE(V7)
{
  int pinValue7 = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(p7,pinValue7);
  delay(100);
  // process received value
}
```

```
void setup()
{
  // Debug console
 Serial.begin(115200);
 pinMode(p0,OUTPUT);
 pinMode(p1,OUTPUT);
 pinMode(p2,OUTPUT);
 pinMode(p3,OUTPUT);
 pinMode(p4,OUTPUT);
 pinMode(p5,OUTPUT);
 pinMode(p6,OUTPUT);
 pinMode(p7,OUTPUT);


  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  // You can also specify server:
  //Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, IPAddress(192,168,1,100), 8080);
}

void loop()
{
  Blynk.run();
}
```

For Forklift:

```
int in1=2;

int in2=3;
```

```
int in3=4;
int in4=5;

int l_fork=10;
int r_fork=9;

int up;
int down;

void setup() {
  // put your setup code here, to run once:
pinMode(in1,OUTPUT);
pinMode(in1,OUTPUT);
pinMode(in3,OUTPUT);
pinMode(in4,OUTPUT);

pinMode(l_fork,INPUT);
pinMode(r_fork,INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
```

```
up=digitalRead(l_fork);
down=digitalRead(r_fork);

if(up==HIGH){

  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in4,HIGH);
  digitalWrite(in3,LOW);
  }
if(up==LOW){
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
  }

if(down==HIGH){

 digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
```

```
  }
if(down==LOW){
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
  }



}
```

Remote Controlled:

```
int startt = 10;
int forward = A1;
int back = 12;
int left = 13;
int right = A0;
int tham = A2;


int left_ir = 8;
int right_ir = 7;
#define MOTOR_PIN_1 2
#define MOTOR_PIN_2 3
#define MOTOR_PIN_3 4
#define MOTOR_PIN_4 5

#define trigPin 6
```

```
#define echoPin 9
long duration;
long distance;

void setup() {
  Serial.begin(9600);
  pinMode(left_ir, INPUT);
  pinMode(right_ir, INPUT);
  pinMode(MOTOR_PIN_1, OUTPUT);
  pinMode(MOTOR_PIN_2, OUTPUT);
  pinMode(MOTOR_PIN_3, OUTPUT);
  pinMode(MOTOR_PIN_4,  OUTPUT);
  pinMode(trigPin,OUTPUT);
 pinMode(echoPin,INPUT);
 pinMode(startt,INPUT);
 pinMode(forward,INPUT);
 pinMode(back,INPUT);
 pinMode(left,INPUT);
 pinMode(right,INPUT);
    pinMode(tham,INPUT);


}

void loop() {

 digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;

 //  Serial.print("distance ");
//Serial.println(distance);
  delay(200);

 int f=analogRead(forward);
  int re=digitalRead(back);
  int l=digitalRead(left);
  int ri=analogRead(right);
```

```
  int ruk=analogRead(tham);

 int l_ir = digitalRead(left_ir);
 int r_ir = digitalRead(right_ir);
//Serial.print("l_ir");
//Serial.println(l_ir);
// Serial.print("r_ir");
// Serial.println(r_ir);

 int shift = digitalRead(startt);
  Serial.print("shift");
  Serial.println(shift);

if(shift== 1){
  // Determine direction based on sensor readings
  if (l_ir == 0 && r_ir == 0 && distance>12 ) {
    // Both sensors on the line, go straight
    analogWrite(MOTOR_PIN_1, 130);
    analogWrite(MOTOR_PIN_2, 0);
    analogWrite(MOTOR_PIN_3, 0);
    analogWrite(MOTOR_PIN_4, 130);
  }
  if (l_ir == 1 && r_ir == 0 && distance>12) {
    // Left sensor on the line, turn left
    analogWrite(MOTOR_PIN_1, 0);
    analogWrite(MOTOR_PIN_2, 130);
    analogWrite(MOTOR_PIN_3, 0);
    analogWrite(MOTOR_PIN_4, 130);
  }
  if (r_ir == 1 && l_ir == 0  && distance>12) {
    // Right sensor on the line, turn right
     analogWrite(MOTOR_PIN_1, 130 );
    analogWrite(MOTOR_PIN_2, 0);
    analogWrite(MOTOR_PIN_3,  130 );
    analogWrite(MOTOR_PIN_4, 0);
  }
  if(l_ir == 1 && r_ir == 1){
    // Both sensors off the line, stop
    analogWrite(MOTOR_PIN_1, 0);
    analogWrite(MOTOR_PIN_2, 0);
    analogWrite(MOTOR_PIN_3, 0);
    analogWrite(MOTOR_PIN_4, 0);
```

```
 }
delay(200);
}
 if(shift==0){
 remote();
  }
}

void remote(){
int f=analogRead(forward);
 int re=digitalRead(back);
 int l=digitalRead(left);
 int ri=analogRead(right);
 int ruk=analogRead(tham);
 Serial.println(ruk);
 if(f>690){
  analogWrite(MOTOR_PIN_1, 130);
  analogWrite(MOTOR_PIN_2, 0);
  analogWrite(MOTOR_PIN_3, 0);
  analogWrite(MOTOR_PIN_4, 130);
 }
  if(f<690){

 }

 if(ri>690){
  analogWrite(MOTOR_PIN_1, 130 );
  analogWrite(MOTOR_PIN_2, 0);
  analogWrite(MOTOR_PIN_3, 130 );
  analogWrite(MOTOR_PIN_4, 0);
 }
  if(ri<690){

 }

 if(l==1){
  analogWrite(MOTOR_PIN_1, 0);
  analogWrite(MOTOR_PIN_2, 130);
  analogWrite(MOTOR_PIN_3, 0);
  analogWrite(MOTOR_PIN_4, 130);
 }
```

```
 if(l==0){

}
 if(re==1){
 analogWrite(MOTOR_PIN_1, 0);
  analogWrite(MOTOR_PIN_2, 130);
  analogWrite(MOTOR_PIN_3, 130);
  analogWrite(MOTOR_PIN_4, 0);
}
 if(re==0){

}
if(ruk>690){
 analogWrite(MOTOR_PIN_1, 0);
  analogWrite(MOTOR_PIN_2, 0);
  analogWrite(MOTOR_PIN_3, 0);
  analogWrite(MOTOR_PIN_4, 0);
}
 if(ruk==0){

}


}
```