AUTOMATIC STAIR CLIMBER FOR PHYSICALLY CHALLENGED PEPOPLE

A PROJECT REPORT

Submitted by

SRAVAN KUMAR T	211420205152
NITHIN CHOWDARY T	211420205100
RITHEECK DHARAN M	211420205125

in partial fulfillment for the award of the degreeof

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report " Automatic Stair Climber For Physically Challenged People" is the bonafide work of "SRAVAN KUMAR T (211420205152), NITHIN CHOWDARY T (211420205152), RITHEECK DHARAN M (211420205125) who carried out the project under my supervision.

SIGNATURE OF HOD
DR.M.HELDA MERCY M.E., Ph.D.,
Department of Information Technology
Panimalar Engineering College
Chennai-123

SIGNATURE OF SUPERVISOR
MRS. K. LALITHA M.E., Ph.D.,
Department of Information Technology
Panimalar Engineering College
Chennai-123

Certified that the above mentioned students were examined in the university project viva-voice held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I here by declare that the project report entitled "**AUTOMATIC STAIR CLIMBER FOR PHYSICALLY CHALLENGED PEOPLE**" which is being submitted in partial fulfilment of the requirement of the course leading to the award of the "Bachelor of Technology in Information Technology" in **Panimalar Engineering College, An Autonomous Institution Affiliated to Anna University-Chennai** is the result of the project carried out by me under the guidance and supervision of **Mrs. K.Lalitha, M.E.,(Ph.D) Associate Professor in the Department of Information Technology.** I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree / diploma or any other person.

(Sravan Kumar T)

(Nithin Chowdary T)

(Ritheeck Dharan M)

Date:

Place: Chennai

It is certified that this project has been prepared and submitted under my guidence.

Date:

Place: Chennai

Mrs. K.Lalitha M.E.,(Ph.D)

(Associate Professor / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many,for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr.P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYA SREE SAKTHIKUMAR., B.E., M.B.A.,Ph.D** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our staff in charge, **Mrs.K.LALITHA M.E.,(Ph.D)** Associate Professor, Department of Information Technology for her guidance throughout the course of our project. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

This task presents the plan and improvement of an IoT-based Robotized Crossover Step Climber taking special care of the portability needs of truly tested people. The framework coordinates trend setting innovations to make a flexible and easy to understand answer for exploring steps easily. An IoT module, using microcontroller innovation and network modules, empowers controller through a committed cell phone application or a controller gadget. Wellbeing highlights, for example, crisis stop buttons and deterrent location sensors, focus on client security. Usefulness incorporates versatile moving with flexible speed, strength guaranteed by gyrator and accelerometer sensors, and a keen slowing mechanism for controlled drop. This task not just addresses the quick versatility challenges looked by truly tested people yet additionally stresses progressing enhancements through IoT availability, guaranteeing the gadget stays versatile to client necessities and wellbeing prerequisites.

CHAPTER 1

INTRODUCTION

The proposed framework is a creative arrangement intended to upgrade the versatility and openness of truly tested people by giving an IoT-based Computerized Cross breed Step Climber. This framework plans to address the difficulties looked by clients in exploring steps, offering a flexible, easy to understand, and mechanically progressed versatility help. The Mechanized Crossover Step Climber coordinates front line mechanical, electrical, and programming parts to make a flexible, easy to use, and safe versatility arrangement. By empowering people to explore steps easily, we mean to add to a more impartial and open climate. The half and half nature of this climber, mixing mechanization with client control, guarantees a customized and versatile experience.

1.1 INTRODUCTION TO STAIR CLIMBING ROBOT

Stairways are omnipresent in man-made environments. These were designed to easily bridge large vertical distances for humans. However, stairs represent a serious challenge to vehicles and robots during the time of disaster such as fire, earthquakes. There is a strong demand for mobile robots that can climb the stairs, for example, to aid people who have difficulty in walking, in urban search and rescue or urban reconnaissance. However, there are few robots that are suitable for use in rough terrains. Most of the existing surface locomotion concepts are based on wheels, caterpillars or legs and have not much evolved lately. Each classification of mobile robot possesses their unique advantages and suffers from certain disadvantages. For the legged robots, they have the capability to adapt to many kinds of unstructured environment and in doing so they can stabilize themselves as different legs can orient themselves with independent configuration. Nonetheless, these robots are instinctively complex and are comparatively slow. The wheeled robot can relate for

the slow locomotive speeds of legged robots as they can move faster because of their rolling motion. However in unstructured conditions, their mobility is often very inadequate and highly depends on the type of surroundings and the typical size of encounter obstacle.

To have a platform with legs that are able to strategically choose contact points on the ground is a vast advantage over wheels in many ways. Not only because of the previously mentioned reason that it can step over obstacles, but also for the fact that it can move smoothly over terrain . Consider a statically stable robot that moves one leg at the time and gently places it at a new stable position, the main body of such a robot would move forward smoothly like a boat, even on really rough terrain like in a forest. The tracked mobile robots have high off-road capability yet ordinarily have overwhelming weight. However, the tracked mobile robots have low energy efficiency in turning motions. On the other hand, the legged mobile robots have great adaptability in rough terrain but usually involves a complex locomotive mechanisms which needs complicated control algorithms. The wheel has always been the easiest way to implement mobility in a vehicle, and also the fastest method of travel. Relative to speed it is also the most energy efficient way to travel. The implementation is often very simple, and does not require any advanced techniques such as vector controllers or additional joints to get the robot moving .The locomotion of all wheeled robots can be primarily categorized as active and passive locomotion. Passive locomotion is a concept based on passive suspensions which involves no sensors or any additional actuators and at the same time guarantees stable movement. Whereas, an active robot generally has an entrenched closed loop control this maintain the solidity of the system during motion.

1.2 FEATURES AND BENEFITS OF STAIR CLIMBING ROBOT

- High stair climbing speed: Capable of achieving speeds of up to 8km/hr and a climbing speed of 4.3 body lengths/second
- Lightweight and rugged design for increased portability
- Platform can be outfitted with an array of sensor capabilities
- More portable than the Pack bot
- Capable of sustained motion for extended periods of operation

1.3 OBJECTIVE

The objective of this work is to first develop a wheeled-leg robot with the capability of climbing stairs with a large variation of height. The high- torque of the motors driving the wheels provide a fast climbing ability of the robot with a robust mechanical design which is capable of enduring high stresses on the uneven ground. The structure of the robot is based on a legged-wheels concept, which has small leg attached to the circumference of the wheel. These legs serves the same purpose as that of the gear, i.e., mating with the next stair step while climbing and pushing the robot to climb to the next step as the wheel rotates. The use of rubber treads on the contact surface of the wheel provide additional grip between the tire and the ground. The rubber layering also provides a mild damping effect. The independent roll of the front and rear wheels adds the much needed capability of overcoming obstacles of the four wheels independently. Such a design enables mobility over a considerable variation in terrains, including hills, rocks and sand.

CHAPTER 2

LITERATURE REVIEW

Moh Myint Maung, Soe Nay Lynn Aung , ”Stair Climbing Robot with Stable Platform, “International Journal of Trend in Scientific Research and Development (IJTSRD) Volume: 3 | Issue: 4 | May-Jun 2019 .

Moh Myint Maung Presents the design and implementation of a remote controlled stair-climbing robot with stable platform. The robot movement is controlled using Arduino UNO and Android Bluetooth connection. The paper presents a complete integrated control design and communication strategy for Bluetooth range. Moreover self-balanced stable platform is designed using MPU6050 IMU sensor. Its mechanical design using DC geared wheels and servo based arm is designed for climbing stairs. The robot system is implemented by using Arduino IDE and MIT App Inventor for android application is developed for remote access. Experimental tests showed that stair climbing process and stable platform were successful integrated and they can be directly applied for various types of stairs and carrying light weighted cap or goods.

Akash Asalekar, Akash Chorage, Bhushan Jagdale, Rohit Kusalkar, Shantanu Garud, Vipul Gaikwad, “Design and Fabrication of Staircase Climbing Robot ”International Research Journal of Engineering and Technology (IRJET),Volume: 04 Issue: 07 | July -2017 .

Staircase climbing robot are important for conducting scientific analysis of objectives. Current mobility designs are complex, using many wheels or legs. An eight wheeled rover capable of traversing rough terrain using an efficient high degree of mobility suspension system. The primary mechanical feature of the stair case climbing mechanism design is its simplicity. Which is accomplish by using only two motors for mobility. Both motors are located inside the body where thermal variations

and disturbance is kept to minimum, increasing the reliability and efficiency. Eight wheels' stair case climbing design robot is used because stability purpose.

Tumula Mani Kota Rajasekhar, M.Sugadev, "Arduino Controlled Special Stair Climbing Wheel Chair Bot", International Journal of Pure and Applied Mathematics Volume 118 No. 24,2018.

Tumula Mani Kota Rajasekhar et.al presents the structure, construction and application of an Arduino controlled special stair climbing robot attached with a wheel chair with which the person sitting in the chair directly climbs the steps. There is a special triangular assembly of three wheels attached to the geared heavy duty DC motors and this whole architecture is controlled with Arduino uno microcontroller board. The proposed stair climbing wheel chair bot can be controlled by a smart phone. The proposed system is very smooth and more comfortable for the person using it. The person sits on it can easily control the system with his mobile phone. This system avoids another human assistance .i.e. the one who sits on it can control it without any others help. This system is a cost effective one

Ramachandran N, Neelesh kumar N , et.al developed a semi-automated lawnmower which may be a replacement for commercial manual lawnmower that is being used widely .This device doesn't require any manpower for their operation, it can be operated by an android device. Commercial lawnmowers are bulkier and are not compatible, but this device is compatible and light-weight. Since it is made up of UPVC(Un-plasticized polyvinyl chloride).Generally alternating current or fuels are used as a power source for commercial lawnmowers whereas this device can be operated by means of battery or solar source.

Dipin.A and Dr. Chandrasekhar.T.K, developed an autonomous Stair climbing robot that will allow the user to the ability to cut their grass with minimal effort. Unlike other robotic Stair climbing robots on the market, this design requires no perimeter wires to maintain the robot within the lawn and also with less human effort

in the manual mode operation. There are some preset pattern installed in the robot, in the automatic mode operation no human effort needed for the operation and helps to cut different patterns in the lawn very easily with less time.. Through an array of sensors safety takes major consideration in the device, this robot will not only stay on the lawn, it will avoid and detect objects and humans. And also it detect the land boundaries and start mowing upon the predefine pattern with the help of installed camera and MATLAB programming.

Puneet Kaushik , Mohit Jain et.al presents a cost-efficient robot that could be used even by people who won't be able to enjoy luxury to use roomba, scooba etc. There is variety of autonomous robots available in the market and they all do their work perfectly according to their specifications but none of them are cost-efficient. For a developing country like India, where the majority of the population is economic, keeping this in mind the robot is designed.

K Aruna Manjusha, S V S Prasad, B.Naresh et.al describes a smart floor cleaning robot that allows cleaning the floor by giving instructions to the robot. This robot makes floor cleaning process easy and fast utilizing a wireless robotic cleaning system. This wireless system consists of a transmitter application that runs on an android mobile app which allows the robot to follow commands given by the user through the transmitter app. The proposed robot consists of Arduino UNO controller which has fourteen digital input/output pins, robotic arm with cleaning pad with a water sprayer for efficient cleaning. The Arduino UNO, on receiving the commands from android device through Bluetooth receiver, decodes the given commands and controls the motors to achieve the desired path and direction.

Manya Jain, Pankaj Singh Rawat et.al details the development of Automatic Floor Cleaner. The project is used for domestic and industrial purpose to clean the surface automatically. When it is turned ON, it sucks in the dust by moving all around the surface (floor or any other area) as it passes over it. The controller is used to drive

the motors and the suction unit also a couple of sensors are used to avoid the obstacles. This can be useful in improving the lifestyle of mankind.

Hala Jamal, Dr. Salih Al-Qaraawi build an obstacle avoidance robotic vehicle using ultrasonic sensors for its movement. A microcontroller (ATmega328) is used to achieve the desired operation. A robot is a machine that can perform task automatically or with guidance. The project proposes robotic vehicle that has an intelligence built in it such that it directs itself whenever an obstacle comes in its path. This robotic vehicle is built, using a micro-controller of AT mega 328 family. An ultrasonic sensor is used to detect any obstacle ahead of it and sends a command to the micro-controller. Depending on the input signal received, the micro-controller redirects the robot to move in an alternate direction by actuating the motors which are interfaced to it through a motor driver

Pavithra A C, Subramanya Goutham V proposes a method for obstacle avoidance of a static obstacle that block the path of the mobile's robot navigation, the proposed method intended to make the robot return the original path of the navigation. Three ultrasonic have been used to measure the distance to the obstacles from three sides of robot, as well as two optical encoders attached to each wheel of wheeled mobile robot is used to count the number of steps that the robot moves along the obstacle and thus find the length of obstacle to avoid it and return the original path to complete the remained distance.

CHAPTER 3

SYSTEM ANALYSIS

3.1 SYSTEM DESCRIPTION

This proposed system represents stair climbing wheel chair bot which can be controlled by android application with Bluetooth interface. This project is mainly proposed to make the life of senior citizens and physically challenged persons easy. This project includes a wheel chair with a triangular wheel alignment connected to heavy load dc motors which are controlled with android application. With this project we can make a wheel chair to climb the staircase easily. Here we use mainly Arduino UNO (ATMEGA 328P), Bluetooth module (HC-05), heavy load dc motor, triangular architecture. The Bluetooth module can be interfaced with the system so that we can easily control the system by smart phone application. This project is more necessary especially in the case of senior citizens and physically challenged persons. The project aims in designing a stair climbing wheel chair that can be operated 3 International Journal of Pure and Applied Mathematics Special Issue using Android mobile phone. The controlling of this is done wirelessly through Android smart phone using the Bluetooth feature present in it. Here in the system the Android smart phone is used as a remote control for operating. The controlling device of the whole system is a Microcontroller. Bluetooth module, DC motors which are interfaced to the Microcontroller. The data received by the Bluetooth module from Android smart phone is fed as input to the controller. The controller acts accordingly get control on the DC motors which helps in climbing the steps. In achieving the task the controller is loaded with a program written using Embedded C language. In this proposed system we are going to design a wheel chair which can be controlled by a smart phone and can also climb the steps. This can be performed with the help of the special triangular architecture arranged in a special way such that, this system overcomes all the disadvantages of the existing system. Here heavy load DC motors

can be used to make the system climb the steps and stepper motor is used to make the system make its turnings. The final model of the base of the system looks like

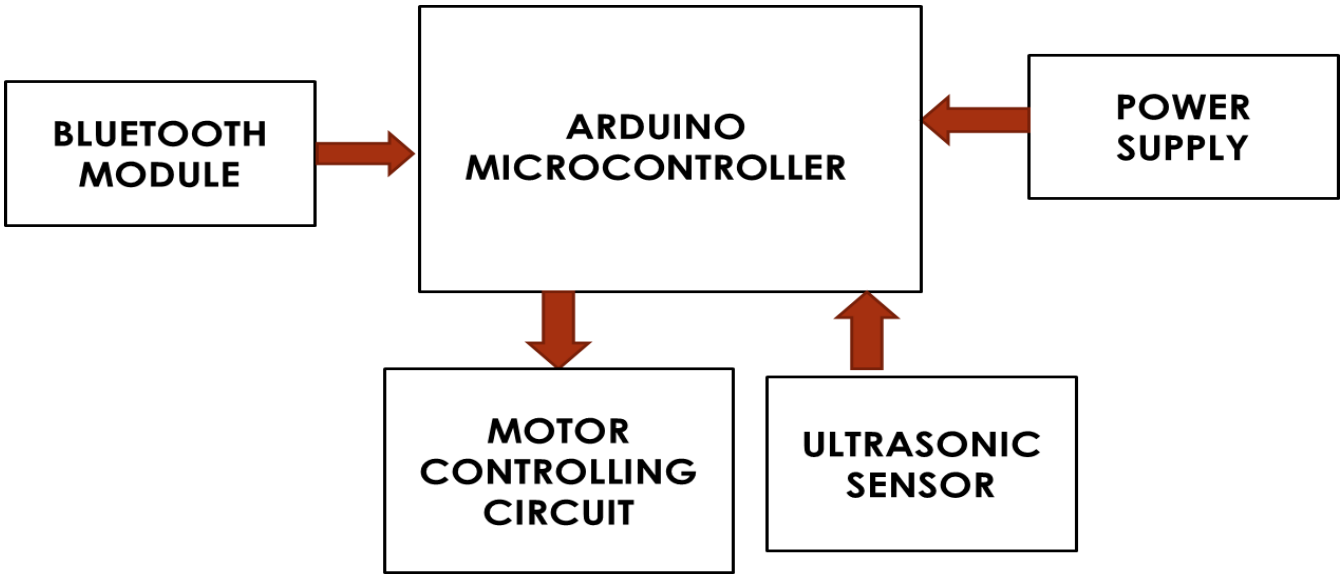


FIG 3.1 BLOCK DIAGRAM

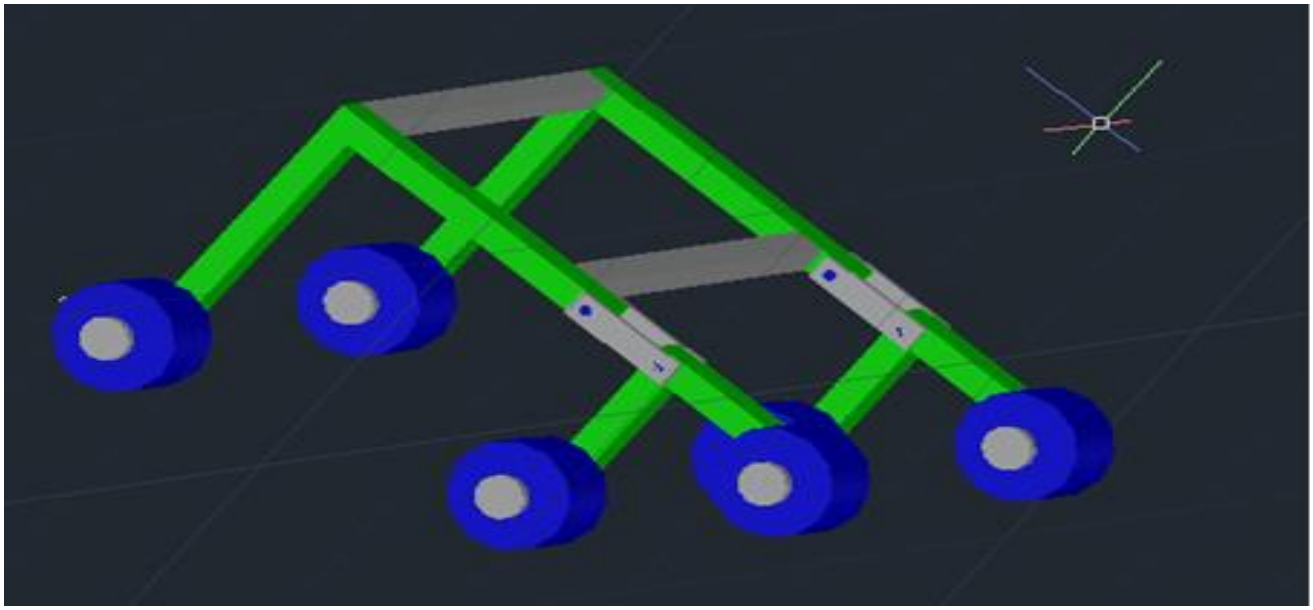


FIG 3.2 2D CAD DESIGN

3.2 METHODOLOGY

The UGV must essentially be a rover which is capable of all terrain mobility due to various terrain and environmental conditions in which it would be required to operate. The UGV consists of the following major sub-systems:

1. Motor wheel sub-assembly
2. Rocker Bogie mechanism
3. Differential

Rocker bogie mechanism the motor wheel sub-assembly is responsible for powering up and movement of the UGV. Rocker bogie mechanism acts as the suspension of the vehicle and helps in rolling the wheels effectively in any terrain. The Rocker bogie mechanism along with the differential contributes to effective maneuvering of the vehicle and avoiding obstacles. The robotic arm and the end effector are the components that are responsible for the removal of the landmines. The drill bit, drillsaround the landmine and the robotic gripper mounted in the end effector assembly would be capable of removing the landmine safely, without triggering the same.

3.2.1 MODELING

AutoCAD software has been used for the 3-D modelling of the UGV. The initial modelling was performed based on a conceptual thinking. However, later, the model was refined using the values or dimensions obtained from the calculations.

3.2.2 Motor wheel sub-assembly

The motor wheel sub assembly consists of two motors and one wheel. The primary motor is a high torque DC geared motor which is wheel mounted and is responsible for the linear motion. The secondary motor is a mega torque DC Servo motor. This is

mounted at the top of the wheel mounting bracket. This provides the steering capability to the wheels. The secondary motor is mounted only to the front two and rear two wheels. The front two wheels turn in one direction while the rear two wheels turn in the opposite direction which gives a very low turning radius to the UGV In order to overcome vertical obstacle faces, the front wheels are forced against the obstacle by the center and rear wheels which generate maximum required torque. The rotation of the front wheel then lifts the front of the vehicle up and over the obstacle and obstacle overtaken. Those wheels which remain in the middle, is then pressed against the obstacle by the rear wheels and pulled against the obstacle by the front till the time it is lifted up and over. At last, the rear wheelis pulled over the obstacle by the front two wheels due to applying pull force. During each wheel's traversal of the obstacle, forward progress of the vehicle is slowed or completely halted which finally maintain vehicles center of gravity.

As per the research it is found that the rocker bogie system reduces the motion by half compared to other suspension systems because each of the bogie's six wheels has an independent mechanism for motion which is essentially a motor mounted on or inside each wheel, inside so that it does not collide with the external terrain during its operation. The two front and two rear wheels have individual steering systems which allow the vehicle to turn in place as 0 degree turning radius.

3.3 WORKING PRINCIPLE

The rocker-bogie design consisting of no springs and stub axles in each wheel which allows the chassis to climb over any obstacles, such as rocks, ditches, sand, etc. that are up to double the wheel's diameter in size while keeping all wheels on the ground maximum time. As compared to any suspension system, the tilt stability is limited by the height of the centre of gravity and the proposed system has the same.

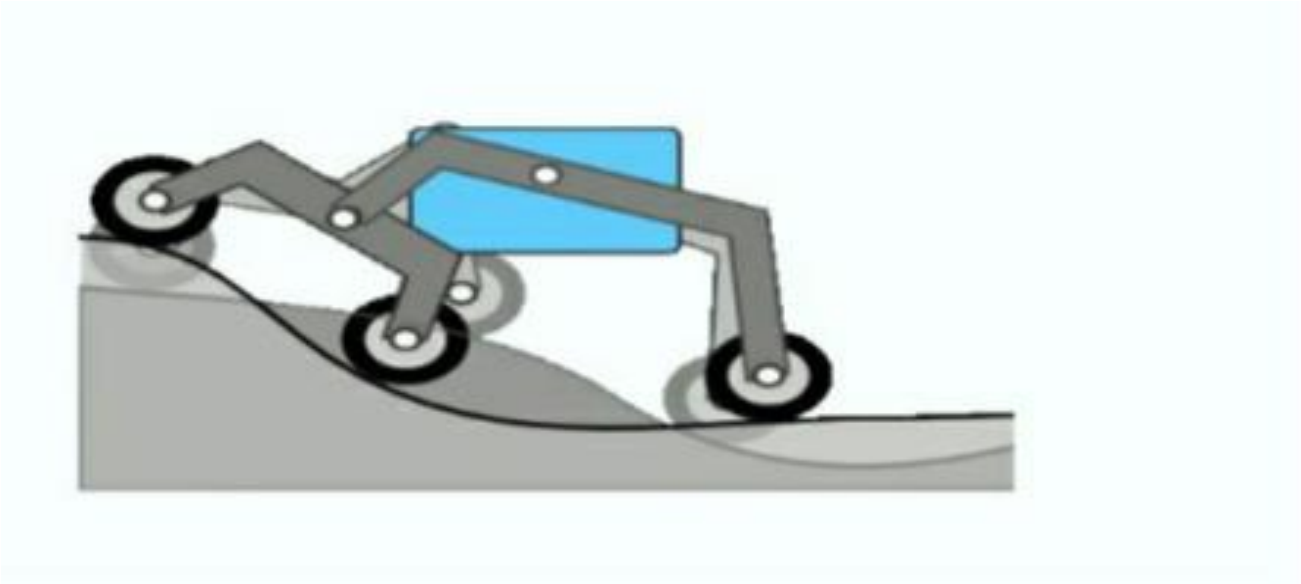


FIG 3.3 IMAGE SHOWING THE MOTION OF THE VEHICLE

Systems employing springs tend to tip more easily as the loaded side yields during obstacle course. Dependent upon the centre of overall weight, any vehicle developed on the basis of Rocker bogie suspension can withstand a tilt of at least 50 degrees in any direction without overturning which is the biggest advantage for any heavy loading vehicle. The system is designed to be implemented in low speed working vehicles such as heavy trucks, Bulldozers which works at slow speed of around 10 centimeters per second (3.9 in/s) so as to minimize dynamic shocks and consequential damage to the vehicle when surmounting sizable obstacles.

CHAPTER 4

HARDWARE IMPLEMENTATION

4.1 POWER SUPPLY

There are many types of power supply. Most are designed to convert the Voltage AC Mains electricity to a suitable low voltage supply for electronic Circuits and other Devices. A power supply can by broken down into a series of blocks, each of which performs a particular function. Here the AC supply main is given to the step down transformer. The transformer having the different voltages.

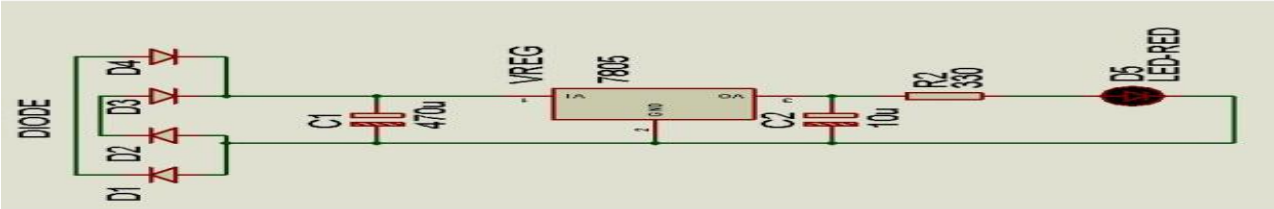


FIG 4.1 CIRCUIT DIAGRAM OF REGULATED POWER SUPPLY

The output from the transformer is given to the rectifier circuit. In this rectifier circuit the AC voltage is converted to DC voltages. The rectified DC voltage is given to the regulator circuit. The output of the regulator is depends upon the regulator IC chosen in the circuit.

4.1.1 BRIDGE RECTIFIER

A bridge rectifier can be made using four individual diodes, but it is also available in special packages containing the four diodes required. It is called a full-wave rectifier. Smoothing is performed by a large value electrolytic capacitor connected across the DC Supply to act as a reservoir, supplying current to the output when the varying DC Voltage from the rectifier is falling.

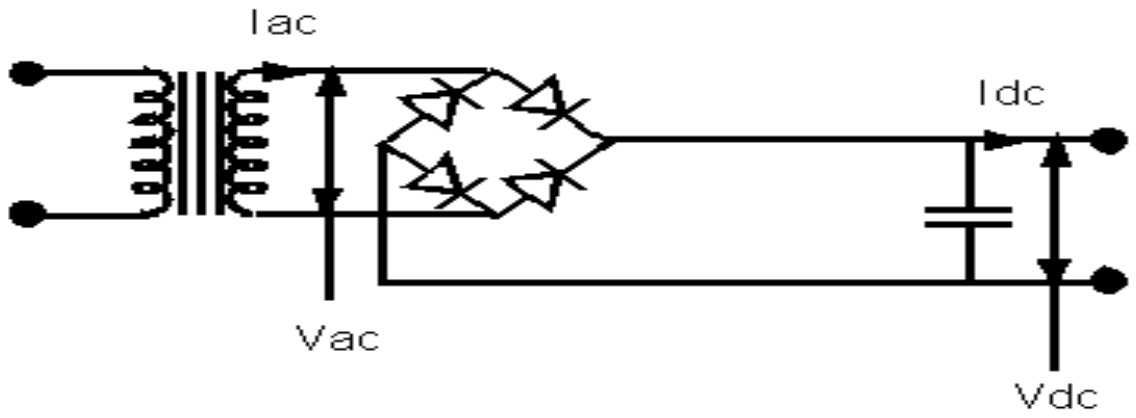


FIG 4.2 BRIDGE RECTIFIER

The fig 4.2 shows the unsmoothed DC, smoothed DC by the filter capacitors. The capacitor charges quickly near the Peak of the varying DC, and then discharges as it supplies current to the output.

Note that smoothing significantly increases the average DC voltage to almost the peak Value (1.4-× RMS value). For example, 6V RMS AC is rectified to full wave DC of about 4.6V RMS (1.4V is lost in the bridge rectifier), with smoothing this increases to almost The peak value giving 1.4 × 4.6 = 6.4V smooth DC. Smoothing is not perfect due to the capacitor voltage falling a little as it discharges, Giving a small ripple voltage. For many circuits a ripple which is 10% of the supply Voltage is satisfactory and the equation below gives the required value for the Smoothing capacitor. A larger capacitor will give less ripple. The capacitor value must Be doubled when smoothing half-wave DC.

4.1.2 REGULATOR

Voltage regulators ICs are available with fixed (typically 5, 12 and 15V) or variable Output voltages. They are also rated by the maximum current they can pass. Negative Voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current (overload protection) and Overheating (thermal protection).

FIG 4.3 REGULATOR

Many of the fixed voltage regulator ICs has 4 leads and look like power transistors, Such as the 7805 +5V 1A regulator shown on the right. They include a hole for attaching a heat sink if necessary.

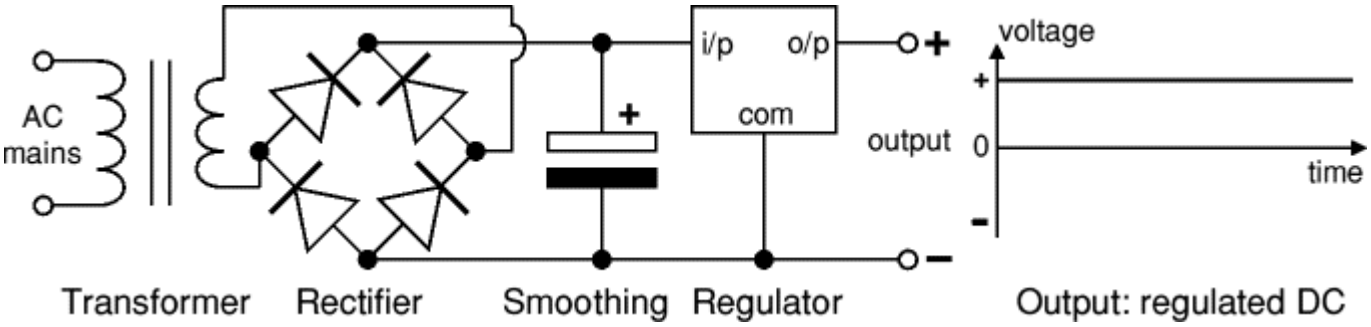


FIG 4.4: RECTIFIER CIRCUIT DIAGRAM AND WAVE FORM

The above fig 4.4 shows the rectifier circuit diagram and the regulated output voltage. The regulated DC output is very smooth with no ripple. In generally there are two types of regulators are used. Namely the positive and negative type regulators. For positive type regulators 78** series of regulators are used. For negative type regulators 79** series of regulators are used. Depends upon the voltage and type of the voltage the regulator IC is selected.

4.2 ARDUINO

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. The Arduino platform has become quite popular with people just

starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

FIG 4.5 ARDUINO

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA

DC Current for 4.4V Pin	50 Ma
Flash Memory	42 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

TABLE 1.1 ARDUINO SUMMARY

4.2.1 ARDUINO PIN DIAGRAM

A typical example of Arduino board is Arduino Uno. It consists of ATmega428- a 28 pin microcontroller. Arduino Uno consists of 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs,a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

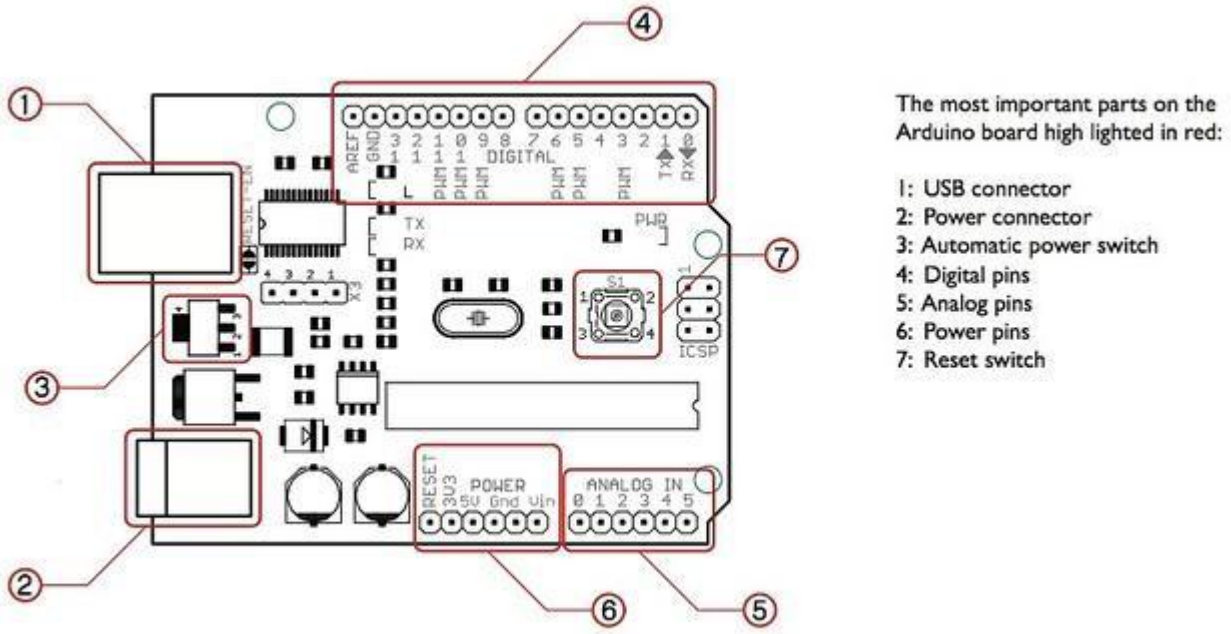


FIG 4.6 ARDUINO PIN DIAGRAM

Power (Usb / Barrel Jack)

Arduino can be power either from the pc through a USB or through external source like adaptor or a battery. It can operate on a external supply of 7 to 12V. Power can be applied externally through the pin Vin or by giving voltage reference through the IORef pin..In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

The USB connection is also how you will load code onto your Arduino board. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5v, 4.4v, Gnd, Analog, Digital, Pwm, Aref)

The pins on your Arduino are the places where you connect wires to construct a circuit probably in conjunction with a breadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- GND : Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- 5V & 4.4V : As you might guess, the 5V pin supplies 5 volts of power, and the 4.4V pin supplies 4.4 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 4.4 volts.
- Analog (5): The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- Digital (4): Across from the analog pins are the digital pins (0 through 14 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- PWM : You may have noticed the tilde (~) next to some of the digital pins (4, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- AREF : Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

1. REST BUTTON

   Just like the original Nintendo, the Arduino has a reset button (7). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn’t repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn’t usually fix any problems.

2. POWER LED INDICATOR

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’. This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong. Time to re-check your circuit.

3. TX RX LEDS

   TX is short for transmit RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for . In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

4.3.1 HC-SR04 ULTRASONIC DISTANCE SENSOR

HC-SR04 Ultrasonic Distance Sensor is a popular and low cost solution for non-contact distance measurement function. It is able to measure distances from 4cm to 400cm with an accuracy of about 4mm. This module includes ultrasonic transmitter, ultrasonic receiver and its control circuit.

HC-SR04 module has 4 pins:

- VCC – 5V, +ive of the power supply
- TRIG – Trigger Pin
- ECHO – Echo Pin
- GND – -ive of the power supply

TRIG and ECHO pins can be used to interface this module with a microcontroller unit. These are TTL (0 – 5V) input output pins.

FIG 4.7 ULTRASONIC SENSOR

4.3.2 HC-SR04 ULTRASONIC MODULE WORKING

Ultrasonic sensors work on a principle similar to sonar which evaluates attributes of a target by interpreting the echoes from sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. The time interval between the sent signal and received signal is determined to measure the distance from an object.

- Provide TRIGGER signal, atleast 10μS High Level (5V) pulse.
- The module will automatically transmit eight 40KHz ultrasonic burst.
- If there is an obstacle in-front of the module, it will reflect the ultrasonic burst.
- If the signal is back, ECHO output of the sensor will be in HIGH state (5V) for a duration of time taken for sending and receiving ultrasonic burst. Pulse width ranges from about 150μS to 25mS and if no obstacle is detected, the echo pulse width will be about 48ms.

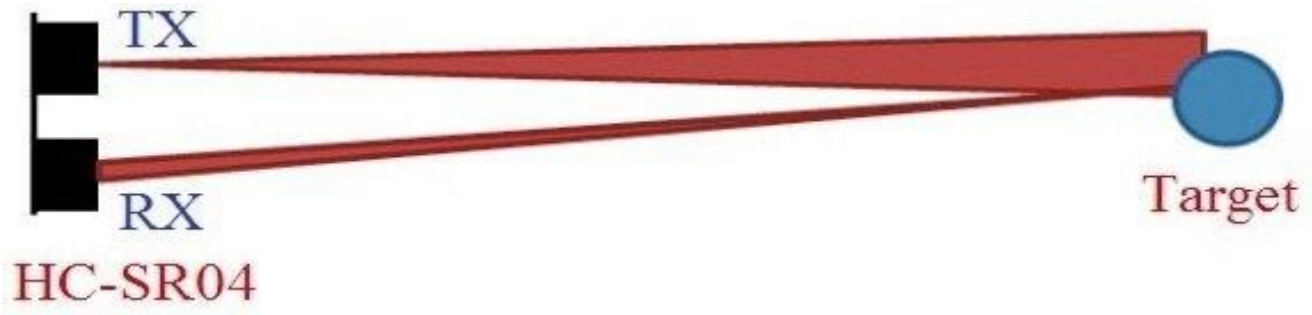


FIG 4.8 WORKING OF HC-SR04 ULTRASONIC SENCOR

4.3.3 DISTANCE MEASUREMENT

The fuel measurement unit comprises of an ultrasonic sensor. The ultrasonic sensor used in our system is HC-SR04 it consist of four pins namely TRIG, ECHO, VCC, GND. This sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. The time interval between the sent signal and received signal is determined to measure the distance from an object. Provide TRIGGER signal, at least 10μS High Level (5V) pulse. The module will automatically transmit eight 40 KHz ultrasonic burst. If there is an obstacle in-front of the module, it will reflect the ultrasonic burst. In our case the obstacle is the fuel. If the signal is back, ECHO output of the sensor will be in HIGH state (5V) for a duration of time taken for sending and receiving ultrasonic burst. Pulse width ranges from about 150μS to 25mS and if no obstacle is detected, the echo pulse width will be about 48ms.

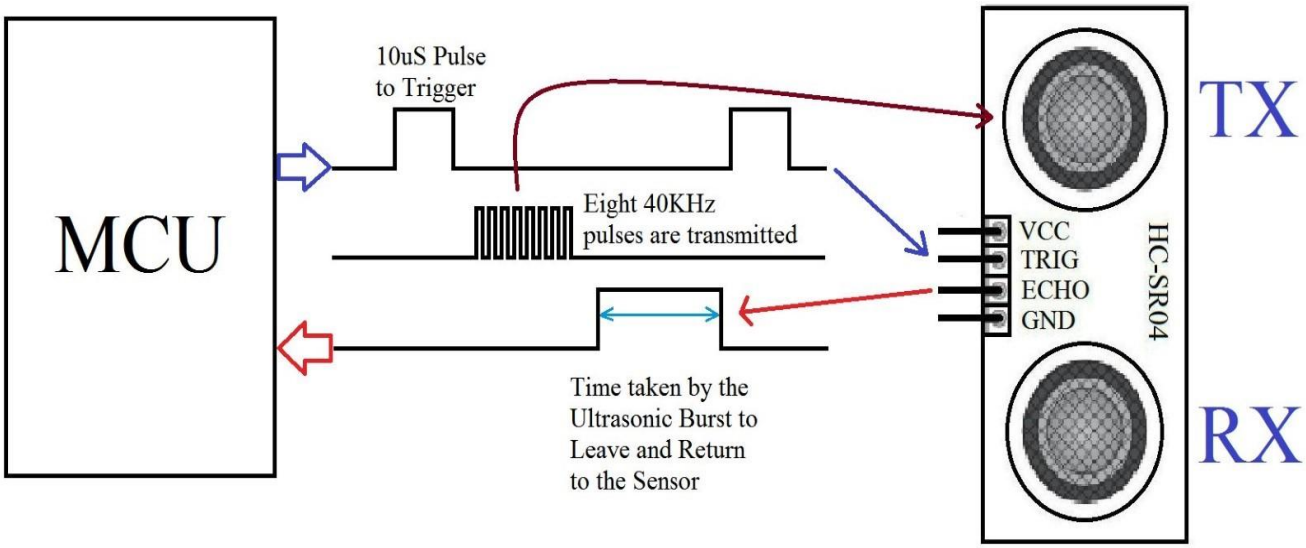


FIG 4.9 OPERATION OF HC-SR04 ULTRASONIC SENSOR

For this purpose we have used an Arduino microcontroller which is programmed to send a high signal to the trig pin for a interval of 10 micro seconds so that the sensor can end eight 40 kHz sound pulse and the time taken between transmission and reception is measured by Arduino as the echo pin is connected to the Arduino. By knowing the time the distance of the liquid can be measured.

4.3.4 TRANSDUCERS FOR WAVE PROPAGATION AND PARTICLE DETECTION

For sending sound waves and receiving echo, ultrasonic sensors, normally called transceivers or transducers will be used. They work on a principle similar to radar that will convert electrical energy into mechanical energy in the form of sound, and vice versa.

The commonly used transducers are contact transducers, angle beam transducers, delay line transducers, immersion transducers, and dual element transducers. Contact transducers are typically used for locating voids and cracks to the outside surface of a part as well as measuring thickness. Angle beam transducers use the principle of

reflection and mode conversion to produce refracted shear or longitudinal waves in the test material.

Delay line transducers are single element longitudinal wave transducers used in conjunction with a replaceable delay line. One of the reasons for choosing delay line transducer is that near surface resolution can be improved. The delay allows the element to stop vibrating before a return signal from the reflector can be received. The major advantages offered by immersion transducers over contact transducers are Uniform coupling reduces sensitivity variations, Reduction in scan time, and increases sensitivity to small reflectors.

4.3.5 ULTRASONIC SENSOR DISTANCE CALCULATION

A pulse is sent for about 10us to trigger the module. After which the module automatically sends 8 cycles of 40 KHz ultrasound signal and checks its echo. The signal after striking with an obstacle returns back and is captured by the receiver. Thus the distance of the obstacle from the sensor is simply calculated by the formula given as **Distance= (time x speed)/2.**

Here we have divided the product of speed and time by 2 because the time is the total time it took to reach the obstacle and return back. Thus the time to reach obstacle is just half the total time taken.

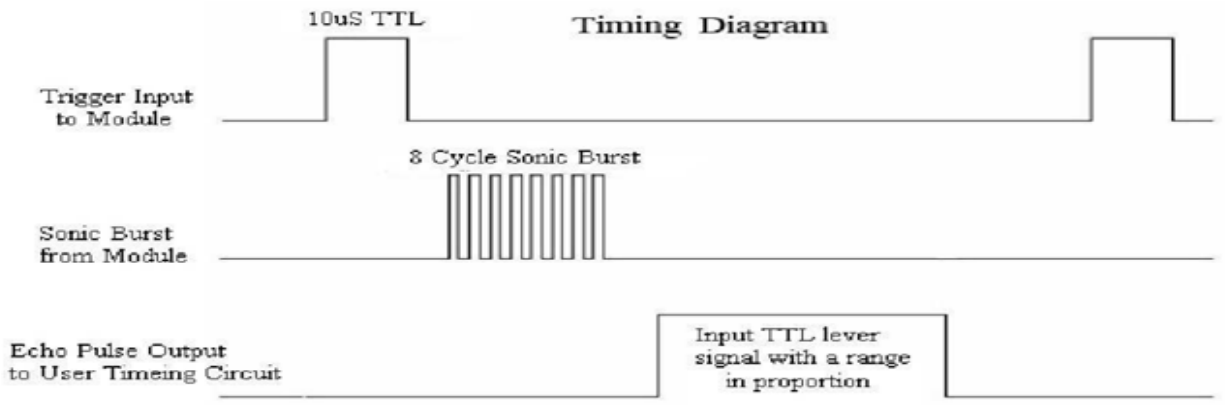


FIG 4.10 TIMING DIAGRAM

First of all we need to trigger the ultrasonic sensor module to transmit signal by using Arduino and then wait for receive ECHO. Arduino reads the time between triggering and Received ECHO. We know that speed of sound is around 340m/s. so we can calculate distance by using given formula:

Distance= (travel time/2) * speed of sound.

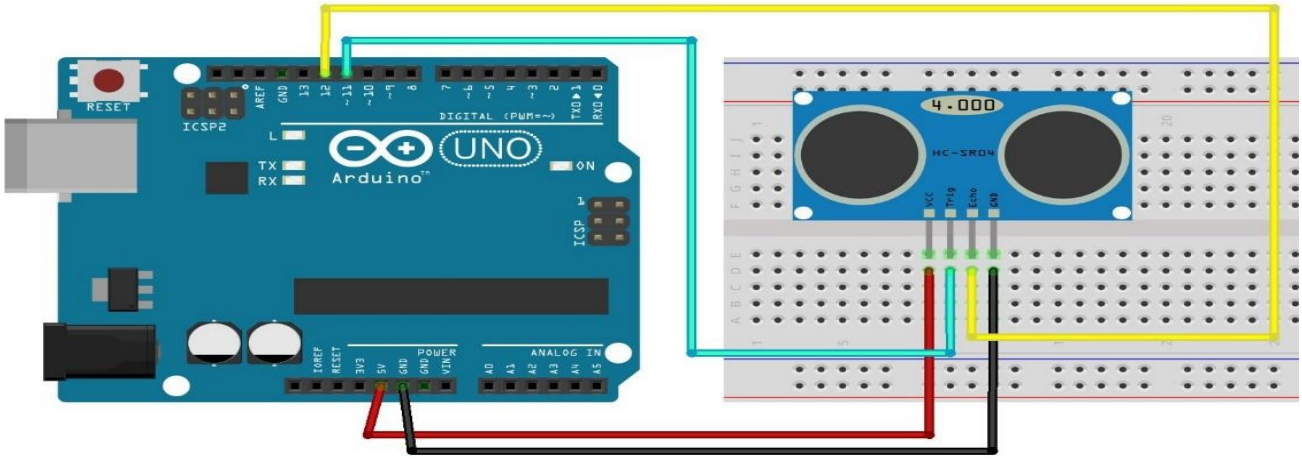
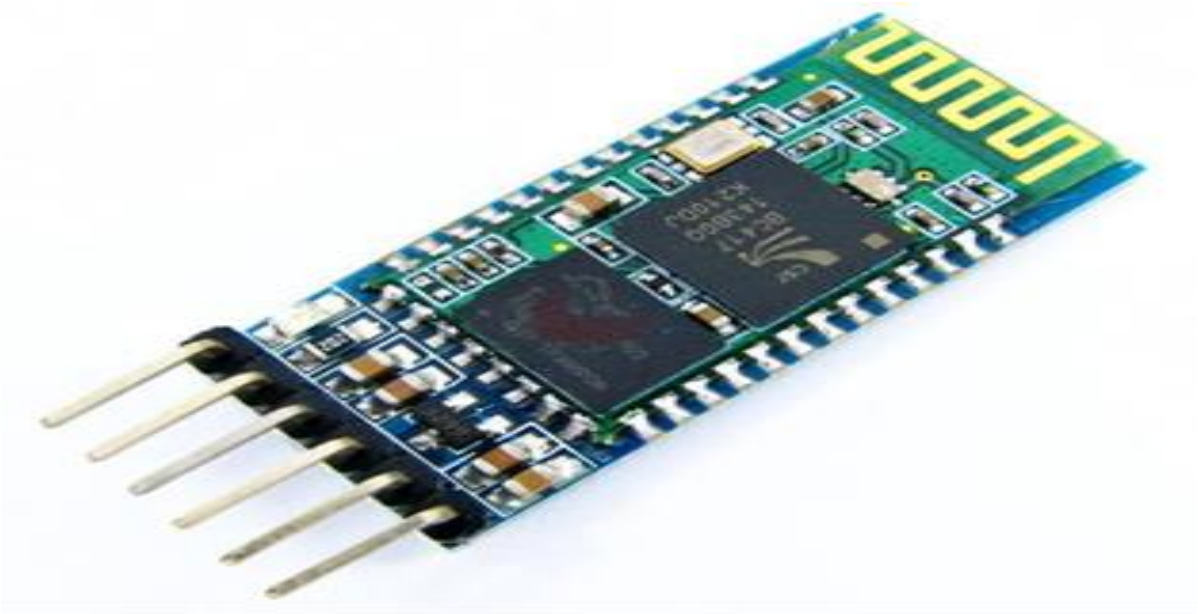


FIG 4.11 CONNECTION OF HC-SR04 ULTRASONIC MODULE WITH ARDUINO MODULE

4.4 HC-05 BLUETOOTH MODULE

HC-05 is a Bluetooth device used for wireless communication with Bluetooth enabled devices (like smartphone). It communicates with microcontrollers using serial communication (USART).It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications. It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions. It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air. It uses serial communication to communicate with devices. It communicates with

microcontroller using serial port (USART).HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration.



4.12 HC-05 BLUETOOTH MODULE

- 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. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.
- As HC-05 Bluetooth module has 3.3 V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.
- Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth. It has 6 pins.

4.13 HC-05 BLUETOOTH PIN DIAGRAM

Key/EN: It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

1. Data mode: Exchange of data between devices.
2. Command mode: It uses AT commands which are used to change setting of HC05. To send these commands to module serial (USART) port is used.
3. VCC: Connect 5 V or 3.3 V to this Pin.
4. GND: Ground Pin of module.
5. TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)
6. RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).
7. State: It tells whether module is connected or not. To communicate smartphone with HC-05 Bluetooth module, smartphone requires Bluetooth terminal application for transmitting and receiving data. You can find Bluetooth terminal applications for android and windows in respective app.

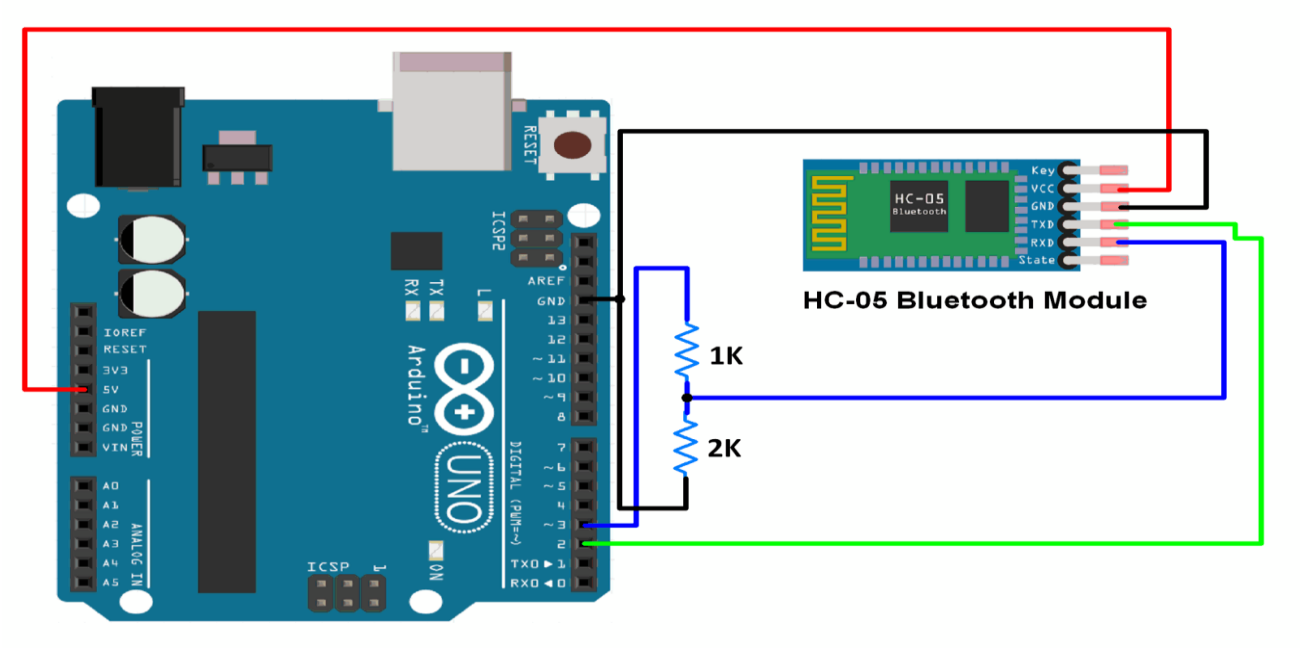
store.Before establishing communication between two Bluetooth devices, 1st we need to pair HC-05 module to smartphone for communication.

Pair HC-05 and smartphone

1. Search for new Bluetooth device from your phone. You will find Bluetooth device with “HC-05” name.
2. Click on connect/pair device option; default pin for HC-05 is 1234 or 0000.
3. After pairing two Bluetooth devices, open terminal software (e.g. Teraterm, Real term etc.) in PC, and select the port where we have connected USB to serial module. Also select default baud rate of 9600 bps.In smart phone, open Bluetooth terminal application and connect to paired device HC-05.
4. It is simple to communicate, we just have to type in the Bluetooth terminal application of smartphone. Characters will get sent wirelessly to Bluetooth module HC-05. HC-05 will automatically transmit it serially to the PC, which will appear on terminal. Same way we can send data from PC to smartphone.

4.4.1 HC-05 BLUETOOTH MODULE INTERFACING

As HC-05 Bluetooth module has 3.3 V level for RX/TX and microcontroller can detect 3.3 V level, so, there is no need to shift TX voltage level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.



4.14 Interfacing HC-05 Bluetooth Module with Arduino UNO

4.5 DC MOTOR DRIVER IC

For the up and down motion of the arm and open close motion of the arm fingers we have used a mechanical arrangement that converts the rotation direction of motor in to equivalent motion. For this purpose we have used an motor driver IC named L293D .

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H-bridge Motor Driver integrated circuit (IC).

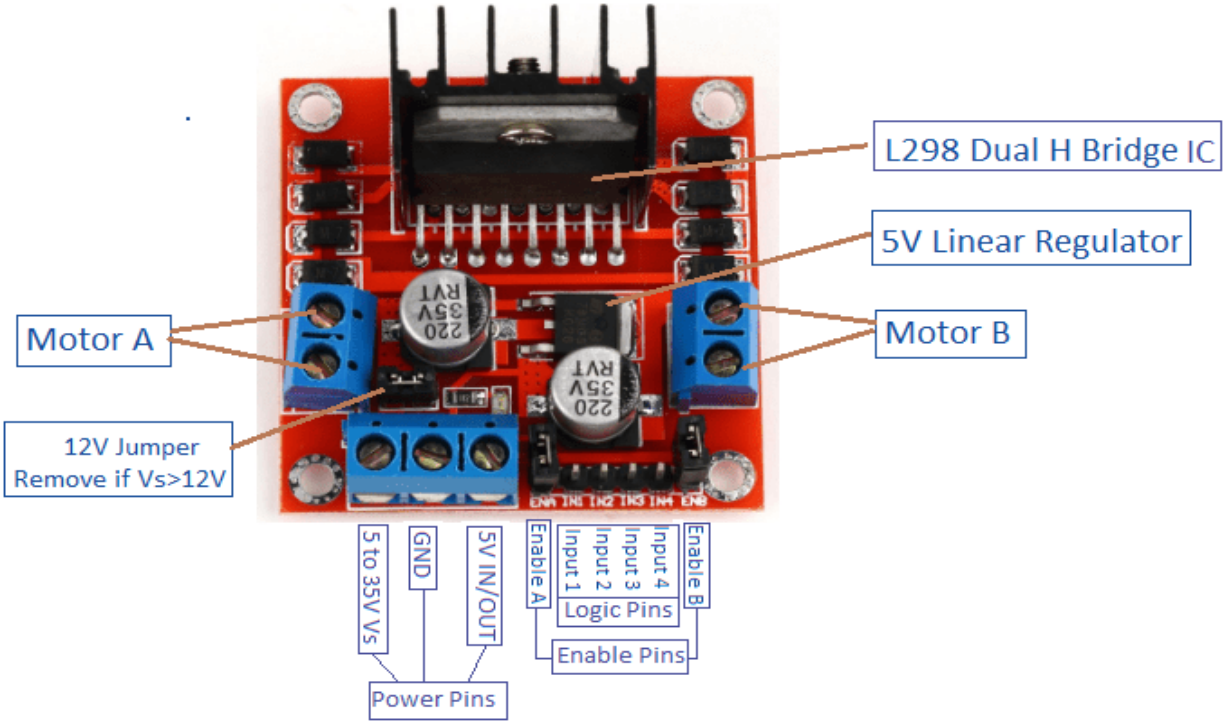


FIG 4.15 L293N MOTOR DRIVER

4.5.1 CONCEPT

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.

In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the

corresponding section will suspend working. It's like a switch.you can simply connect the pin16 VCC (5v) to pin 1 and pin 9 to make them high.

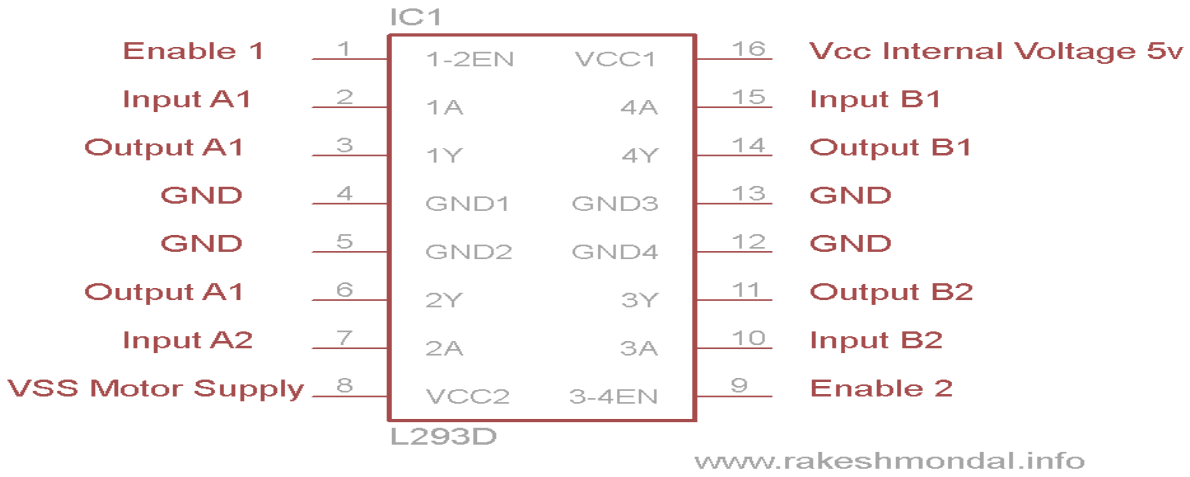


FIG 4.16 L293D PIN DIAGRAM

4.5.2 WORKING OF L293D

There are 4 input pins for l293d, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

4.5.3 L293D LOGIC TABLE

Lets consider a Motor connected on left side output pins (pin 3,6). For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

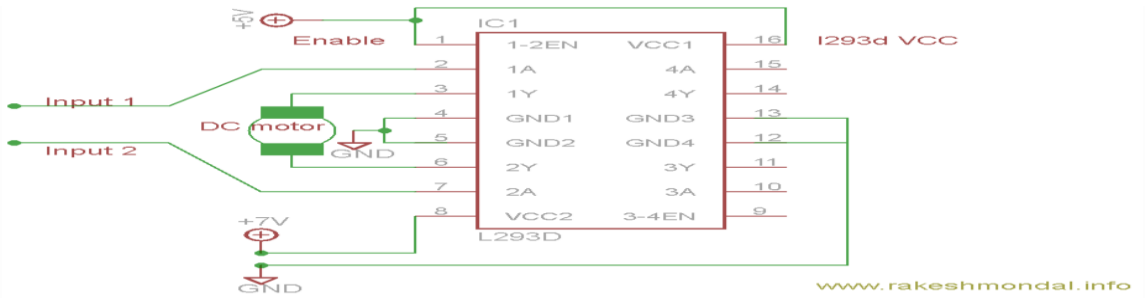


FIG 4.17 Circuit Diagram for l293d motor driver IC controller.

- Pin 2 = Logic 1 and Pin 7 = Logic 0 Clockwise Direction
- Pin 2 = Logic 0 and Pin 7 = Logic 1 Anticlockwise Direction

- Pin 2 = Logic 0 and Pin 7 = Logic 0 | Idle [No rotation] [Hi-Impedance state]
- Pin 2 = Logic 1 and Pin 7 = Logic 1 | Idle [No rotation]In a very similar way the motor can also operate across input pin 15,10 for motor on the right hand side.

4.5.4 VOLTAGE SPECIFICATION

VCC is the voltage that it needs for its own internal operation 5v; L293D will not use this voltage for driving the motor. For driving the motors it has a separate provision to provide motor supply VSS (V supply). L293d will use this to drive the motor. It means if you want to operate a motor at 9V then you need to provide a Supply of 9V across VSS Motor supply.

The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel. Since it can drive motors Up to 36v hence you can drive pretty big motors with this l293d.VCC pin 16 is the voltage for its own internal Operation. The maximum voltage ranges from 5v and up to 36v.

4.5.5 INTERFACING

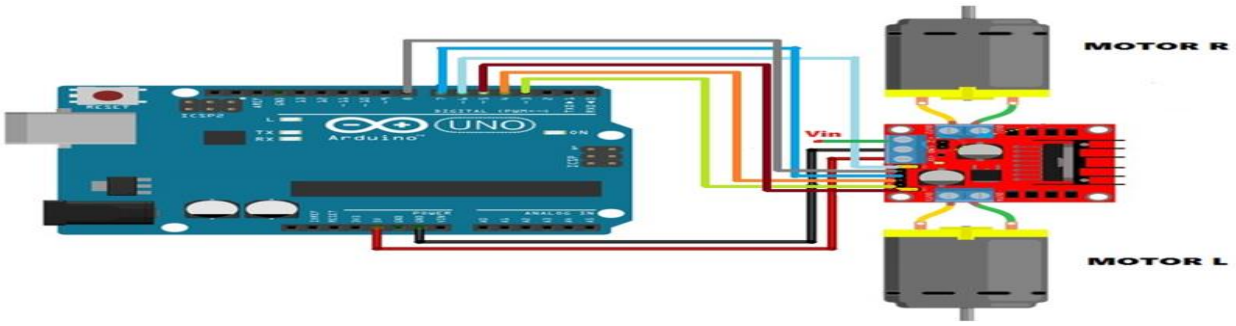


FIG 4.18 MOTOR DRIVER INTERFACING

With L293D Motor Driver IC, we can actually control two motors. For simplicity reasons, I'll demonstrate the circuit, working and program for controlling a single DC Motor ARDUINO. The following image is the Fritzing diagram of the project.

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 ARDUINO 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.

5.2 SOFTWARE OVERVIEW

Programs 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 righthand 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.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.

The following are the basic menu elements present in the Arduino IDE Software for coding, compiling and uploading a sketch on to an Arduino. Each menu has a different icon for easy identification. This icons are shown below along with there operations.

 Verify [Text Wrapping Break]Checks your code for errors compiling it.

 Compiles your code and uploads it to the configured board. See uploading below for details.

 Creates a new sketch.

 Open Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

 Saves your sketch.

 SerialMonitor Opens the serial monitor

5.2.1 PROGRAMMING

When you open the Arduino program, you are opening the IDE. It is intentionally streamlined to keep things as simple and straightforward as possible. When you save a file in Arduino, the file is called a sketch – a sketch is where you save the computer code you have written.

The coding language that Arduino uses is very much like C++ (“see plus plus”), which is a common language in the world of computing. The code you learn to write for Arduino will be very similar to the code you write in any other computer language – all the basic concepts remain the same – it is just a matter of learning a new dialect should you pursue other programming languages.

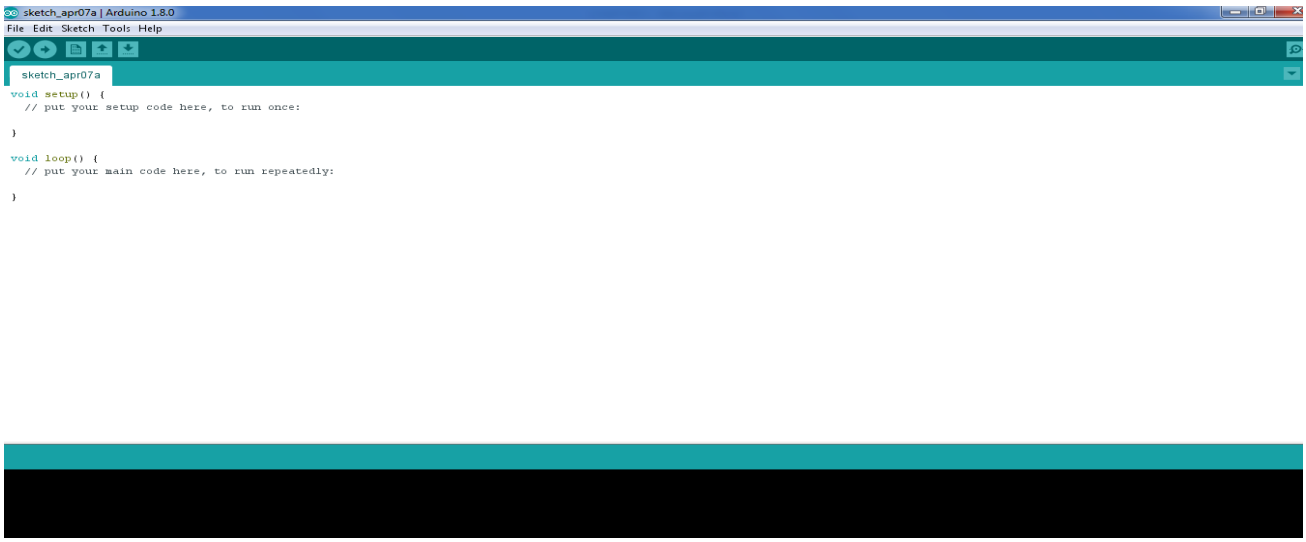


FIG 5.1 PROGRAMMING WINDOW

The code you write is “human readable”, that is, it will make sense to you (sometimes), and will be organized for a human to follow. Part of the job of the IDE is to take the human readable code and translate it into machine-readable code to be executed by the Arduino. This process is called compiling.

The process of compiling is seamless to the user. All you have to do is press a button. If you have errors in your computer code, the compiler will display an error message at the bottom of the IDE and highlight the line of code that seems to be the issue. The error message is meant to help you identify what you might have done wrong – sometimes the message is very explicit, like saying, “Hey – you forget a semicolon”, sometimes the error message is vague.Why be concerned with a semicolon you ask? A semicolon is part of the Arduino language syntax, the rules that govern how the code is written. It is like grammar in writing. Say for example we didn’t use periods when we wrote – everyone would have a heck of a time trying to figure out when sentences started and ended. Or if we didn’t employ the comma, how would we convey a dramatic pause to the reader?And let me tell you, if you ever had an English teacher with an overactive red pen, the compiler is ten times worse. In fact – your

programs WILL NOT compile without perfect syntax. This might drive you crazy at first because it is very natural to forget syntax. As you gain experience programming you will learn to be assiduous about coding grammar.

BlynkEdgent.h

```
extern "C" {
  #include "user_interface.h"

  void app_loop();
}

#include "Settings.h"
#include <BlynkSimpleEsp8266_SSL.h>

#ifndef BLYNK_NEW_LIBRARY
#error "Old version of Blynk library is in use. Please replace it with the new
one."
#endif

#if !defined(BLYNK_TEMPLATE_ID) || !defined(BLYNK_DEVICE_NAME)
#error "Please specify your BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME"
#endif

#include "BlynkState.h"
#include "ConfigStore.h"
#include "ResetButton.h"
#include "ConfigMode.h"
#include "Indicator.h"
#include "OTA.h"
#include "Console.h"

BlynkTimer edgentTimer;

inline
void BlynkState::set(State m) {
  if (state != m && m < MODE_MAX_VALUE) {
    DEBUG_PRINT(String(StateStr[state]) + " => " + StateStr[m]);
    state = m;

    // You can put your state handling here,
    // i.e. implement custom indication
  }
}

void printDeviceBanner()
```

```
{
  Blynk.printBanner();
  DEBUG_PRINT("-------------------------");
  DEBUG_PRINT(String("Product:  ") + BLYNK_DEVICE_NAME);
  DEBUG_PRINT(String("Firmware: ") + BLYNK_FIRMWARE_VERSION " (build " __DATE__ " "
__TIME__ ")");
  if (configStore.getFlag(CONFIG_FLAG_VALID)) {
    DEBUG_PRINT(String("Token:    ...") + (configStore.cloudToken+28));
  }
  DEBUG_PRINT(String("Device:   ") + BLYNK_INFO_DEVICE + " @ " +
ESP.getCpuFreqMHz() + "MHz");
  DEBUG_PRINT(String("MAC:      ") + WiFi.macAddress());
  DEBUG_PRINT(String("Flash:    ") + ESP.getFlashChipRealSize() / 1024 + "K");
  String coreVer = ESP.getCoreVersion();
  coreVer.replace("_", ".");
  DEBUG_PRINT(String("ESP core: ") + coreVer);
  DEBUG_PRINT(String("ESP SDK:  ") + ESP.getSdkVersion());
  DEBUG_PRINT(String("Boot Ver: ") + ESP.getBootVersion());
  DEBUG_PRINT(String("Boot Mode:") + ESP.getBootMode());
  DEBUG_PRINT(String("FW info:  ") + ESP.getSketchSize() + "/" +
ESP.getFreeSketchSpace() + ", MD5:" + ESP.getSketchMD5());
  DEBUG_PRINT(String("Free mem: ") + ESP.getFreeHeap());
  DEBUG_PRINT("-------------------------");
}

void runBlynkWithChecks() {
  Blynk.run();
  if (BlynkState::get() == MODE_RUNNING) {
    if (!Blynk.connected()) {
      if (WiFi.status() == WL_CONNECTED) {
        BlynkState::set(MODE_CONNECTING_CLOUD);
      } else {
        BlynkState::set(MODE_CONNECTING_NET);
      }
    }
  }
}

class Edgent {

public:
  void begin()
  {
    indicator_init();
    button_init();
    config_init();
    edgentTimer.setTimeout(1000L, console_init);

    printDeviceBanner();
```

```
    if (configStore.getFlag(CONFIG_FLAG_VALID)) {
      BlynkState::set(MODE_CONNECTING_NET);
    } else if (config_load_blnkopt()) {
      DEBUG_PRINT("Firmware is preprovisioned");
      BlynkState::set(MODE_CONNECTING_NET);
    } else {
      BlynkState::set(MODE_WAIT_CONFIG);
    }
  }

  void run() {
    app_loop();
    switch (BlynkState::get()) {
    case MODE_WAIT_CONFIG:
    case MODE_CONFIGURING:       enterConfigMode();    break;
    case MODE_CONNECTING_NET:    enterConnectNet();    break;
    case MODE_CONNECTING_CLOUD:  enterConnectCloud();  break;
    case MODE_RUNNING:           runBlynkWithChecks(); break;
    case MODE_OTA_UPGRADE:       enterOTA();           break;
    case MODE_SWITCH_TO_STA:     enterSwitchToSTA();   break;
    case MODE_RESET_CONFIG:      enterResetConfig();   break;
    default:                     enterError();         break;
    }
  }

};

Edgent BlynkEdgent;

void app_loop() {
    edgentTimer.run();
    edgentConsole.run();
}
```

BlynkState.h

```
enum State {
  MODE_WAIT_CONFIG,
  MODE_CONFIGURING,
  MODE_CONNECTING_NET,
  MODE_CONNECTING_CLOUD,
  MODE_RUNNING,
  MODE_OTA_UPGRADE,
  MODE_SWITCH_TO_STA,
```

```
  MODE_RESET_CONFIG,
  MODE_ERROR,

  MODE_MAX_VALUE
};

#if defined(APP_DEBUG)
const char* StateStr[MODE_MAX_VALUE+1] = {
  "WAIT_CONFIG",
  "CONFIGURING",
  "CONNECTING_NET",
  "CONNECTING_CLOUD",
  "RUNNING",
  "OTA_UPGRADE",
  "SWITCH_TO_STA",
  "RESET_CONFIG",
  "ERROR",

  "INIT"
};
#endif

namespace BlynkState
{
  volatile State state = MODE_MAX_VALUE;

  State get()        { return state; }
  bool  is (State m) { return (state == m); }
  void  set(State m);
};
```

ConfigMode.h

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPUpdateServer.h>
#include <DNSServer.h>

ESP8266WebServer server(80);
ESP8266HTTPUpdateServer httpUpdater;
DNSServer dnsServer;
const byte DNS_PORT = 53;
```

```
#ifdef BLYNK_USE_SPIFFS
  #include <FS.h>
#else
  const char* config_form = R"html(
<!DOCTYPE HTML>
<html>
<head>
  <title>WiFi setup</title>
  <style>
  body {
    background-color: #fcfcfc;
    box-sizing: border-box;
  }
  body, input {
    font-family: Roboto, sans-serif;
    font-weight: 400;
    font-size: 16px;
  }
  .centered {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);

    padding: 20px;
    background-color: #ccc;
    border-radius: 4px;
  }
  td { padding:0 0 0 5px; }
  label { white-space:nowrap; }
  input { width: 20em; }
  input[name="port"] { width: 5em; }
  input[type="submit"], img { margin: auto; display: block; width: 30%; }
  </style>
</head>
<body>
<div class="centered">
  <form method="get" action="config">
    <table>
    <tr><td><label for="ssid">WiFi SSID:</label></td>  <td><input type="text"
name="ssid" length=64 required="required"></td></tr>
    <tr><td><label for="pass">Password:</label></td>   <td><input type="text"
name="pass" length=64></td></tr>
    <tr><td><label for="blynk">Auth token:</label></td><td><input type="text"
name="blynk" placeholder="a0b1c2d..." pattern="[-_a-zA-Z0-9]{32}" maxlength="32"
required="required"></td></tr>
    <tr><td><label for="host">Host:</label></td>       <td><input type="text"
name="host" value="blynk.cloud" length=64></td></tr>
```

```
    <tr><td><label for="port_ssl">Port:</label></td>   <td><input type="number"
name="port_ssl" value="443" min="1" max="65535"></td></tr>
    </table><br/>
    <input type="submit" value="Apply">
  </form>
</div>
</body>
</html>
)html";
#endif

void restartMCU() {
  ESP.restart();
  delay(10000);
  ESP.reset();
  while(1) {};
}


void getWiFiName(char* buff, size_t len, bool withPrefix = true) {
  byte mac[6] = { 0, };
  WiFi.macAddress(mac);

  uint32_t unique = 0;
  for (int i=0; i<4; i++) {
    unique = BlynkCRC32(&mac, sizeof(mac), unique);
  }
  unique &= 0xFFFFF;

  String devName = String(BLYNK_DEVICE_NAME).substring(0, 31-6-6);

  if (withPrefix) {
    snprintf(buff, len, "Blynk %s-%05X", devName.c_str(), unique);
  } else {
    snprintf(buff, len, "%s-%05X", devName.c_str(), unique);
  }
}

void enterConfigMode()
{
  char ssidBuff[64];
  getWiFiName(ssidBuff, sizeof(ssidBuff));

  WiFi.mode(WIFI_OFF);
  delay(100);
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAPConfig(WIFI_AP_IP, WIFI_AP_IP, WIFI_AP_Subnet);
  WiFi.softAP(ssidBuff);
  delay(500);
```

```
  IPAddress myIP = WiFi.softAPIP();
  DEBUG_PRINT(String("AP SSID: ") + ssidBuff);
  DEBUG_PRINT(String("AP IP:   ") + myIP[0] + "." + myIP[1] + "." + myIP[2] + "." +
myIP[3]);

  if (myIP == (uint32_t)0)
  {
    config_set_last_error(BLYNK_PROV_ERR_INTERNAL);
    BlynkState::set(MODE_ERROR);
    return;
  }

  // Set up DNS Server
  dnsServer.setTTL(300); // Time-to-live 300s
  dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure); // Return code for non-
accessible domains
#ifdef WIFI_CAPTIVE_PORTAL_ENABLE
  dnsServer.start(DNS_PORT, "*", WiFi.softAPIP()); // Point all to our IP
  server.onNotFound(handleRoot);
#else
  dnsServer.start(DNS_PORT, CONFIG_AP_URL, WiFi.softAPIP());
  DEBUG_PRINT(String("AP URL:  ") + CONFIG_AP_URL);
#endif

  httpUpdater.setup(&server, "/update");

#ifndef BLYNK_USE_SPIFFS
  server.on("/", []() {
    server.send(200, "text/html", config_form);
  });
#endif
  server.on("/config", []() {
    DEBUG_PRINT("Applying configuration...");
    String ssid = server.arg("ssid");
    String ssidManual = server.arg("ssidManual");
    String pass = server.arg("pass");
    if (ssidManual != "") {
      ssid = ssidManual;
    }
    String token = server.arg("blynk");
    String host  = server.arg("host");
    String port  = server.arg("port_ssl");

    String ip   = server.arg("ip");
    String mask = server.arg("mask");
    String gw   = server.arg("gw");
    String dns  = server.arg("dns");
    String dns2 = server.arg("dns2");
```

```
  bool save  = server.arg("save").toInt();

  String content;

  DEBUG_PRINT(String("WiFi SSID: ") + ssid + " Pass: " + pass);
  DEBUG_PRINT(String("Blynk cloud: ") + token + " @ " + host + ":" + port);

  if (token.length() == 32 && ssid.length() > 0) {
    configStore.setFlag(CONFIG_FLAG_VALID, false);
    CopyString(ssid, configStore.wifiSSID);
    CopyString(pass, configStore.wifiPass);
    CopyString(token, configStore.cloudToken);
    if (host.length()) {
      CopyString(host,  configStore.cloudHost);
    }
    if (port.length()) {
      configStore.cloudPort = port.toInt();
    }

    IPAddress addr;

    if (ip.length() && addr.fromString(ip)) {
      configStore.staticIP = addr;
      configStore.setFlag(CONFIG_FLAG_STATIC_IP, true);
    } else {
      configStore.setFlag(CONFIG_FLAG_STATIC_IP, false);
    }
    if (mask.length() && addr.fromString(mask)) {
      configStore.staticMask = addr;
    }
    if (gw.length() && addr.fromString(gw)) {
      configStore.staticGW = addr;
    }
    if (dns.length() && addr.fromString(dns)) {
      configStore.staticDNS = addr;
    }
    if (dns2.length() && addr.fromString(dns2)) {
      configStore.staticDNS2 = addr;
    }

    if (save) {
      configStore.setFlag(CONFIG_FLAG_VALID, true);
      config_save();

      content = R"json({"status":"ok","msg":"Configuration saved"})json";
    } else {
      content = R"json({"status":"ok","msg":"Trying to connect..."})json";
    }
```

```
      server.send(200, "application/json", content);

      BlynkState::set(MODE_SWITCH_TO_STA);
    } else {
      DEBUG_PRINT("Configuration invalid");
      content = R"json({"status":"error","msg":"Configuration invalid"})json";
      server.send(500, "application/json", content);
    }
  });
  server.on("/board_info.json", []() {
    DEBUG_PRINT("Sending board info...");
    const char* tmpl = BLYNK_TEMPLATE_ID;
    char ssidBuff[64];
    getWiFiName(ssidBuff, sizeof(ssidBuff));
    char buff[512];
    snprintf(buff, sizeof(buff),

R"json({"board":"%s","tmpl_id":"%s","fw_type":"%s","fw_ver":"%s","ssid":"%s","bssid
":"%s","mac":"%s","last_error":%d,"wifi_scan":true,"static_ip":true})json",
      BLYNK_DEVICE_NAME,
      tmpl ? tmpl : "Unknown",
      BLYNK_FIRMWARE_TYPE,
      BLYNK_FIRMWARE_VERSION,
      ssidBuff,
      WiFi.softAPmacAddress().c_str(),
      WiFi.macAddress().c_str(),
      configStore.last_error
    );
    server.send(200, "application/json", buff);
  });
  server.on("/wifi_scan.json", []() {
    DEBUG_PRINT("Scanning networks...");
    int wifi_nets = WiFi.scanNetworks(true, true);
    const uint32_t t = millis();
    while (wifi_nets < 0 &&
           millis() - t < 20000)
    {
      delay(20);
      wifi_nets = WiFi.scanComplete();
    }
    DEBUG_PRINT(String("Found networks: ") + wifi_nets);

    if (wifi_nets > 0) {
      // Sort networks
      int indices[wifi_nets];
      for (int i = 0; i < wifi_nets; i++) {
        indices[i] = i;
      }
      for (int i = 0; i < wifi_nets; i++) {
```

```
      for (int j = i + 1; j < wifi_nets; j++) {
        if (WiFi.RSSI(indices[j]) > WiFi.RSSI(indices[i])) {
          std::swap(indices[i], indices[j]);
        }
      }
    }

    wifi_nets = BlynkMin(15, wifi_nets); // Show top 15 networks

    // TODO: skip empty names
    server.setContentLength(CONTENT_LENGTH_UNKNOWN);
    server.send(200, "application/json", "[\n");


    char buff[256];
    for (int i = 0; i < wifi_nets; i++){
      int id = indices[i];

      const char* sec;
      switch (WiFi.encryptionType(id)) {
      case ENC_TYPE_WEP:  sec = "WEP"; break;
      case ENC_TYPE_TKIP: sec = "WPA/PSK"; break;
      case ENC_TYPE_CCMP: sec = "WPA2/PSK"; break;
      case ENC_TYPE_AUTO: sec = "WPA/WPA2/PSK"; break;
      case ENC_TYPE_NONE: sec = "OPEN"; break;
      default:            sec = "unknown"; break;
      }

      snprintf(buff, sizeof(buff),
        R"json(
{"ssid":"%s","bssid":"%s","rssi":%i,"sec":"%s","ch":%i,"hidden":%d})json",
        WiFi.SSID(id).c_str(),
        WiFi.BSSIDstr(id).c_str(),
        WiFi.RSSI(id),
        sec,
        WiFi.channel(id),
        WiFi.isHidden(id)
      );

      server.sendContent(buff);
      if (i != wifi_nets-1) server.sendContent(",\n");
    }
    server.sendContent("\n]");
  } else {
    server.send(200, "application/json", "[]");
  }
});
server.on("/reset", []() {
  BlynkState::set(MODE_RESET_CONFIG);
```

```
    server.send(200, "application/json", R"json({"status":"ok","msg":"Configuration
reset"})json");
  });
  server.on("/reboot", []() {
    restartMCU();
  });

#ifdef BLYNK_USE_SPIFFS
  if (SPIFFS.begin()) {
    server.serveStatic("/img", SPIFFS, "/img");
    server.serveStatic("/", SPIFFS, "/index.html");
  } else {
    DEBUG_PRINT("Webpage: No SPIFFS");
  }
#endif

  server.begin();

  while (BlynkState::is(MODE_WAIT_CONFIG) || BlynkState::is(MODE_CONFIGURING)) {
    delay(10);
    dnsServer.processNextRequest();
    server.handleClient();
    app_loop();
    if (BlynkState::is(MODE_WAIT_CONFIG) && WiFi.softAPgetStationNum() > 0) {
      BlynkState::set(MODE_CONFIGURING);
    } else if (BlynkState::is(MODE_CONFIGURING) && WiFi.softAPgetStationNum() == 0)
{
      BlynkState::set(MODE_WAIT_CONFIG);
    }
  }

  server.stop();

#ifdef BLYNK_USE_SPIFFS
  SPIFFS.end();
#endif
}

void enterConnectNet() {
  BlynkState::set(MODE_CONNECTING_NET);
  DEBUG_PRINT(String("Connecting to WiFi: ") + configStore.wifiSSID);

  WiFi.mode(WIFI_STA);

  char ssidBuff[64];
  getWiFiName(ssidBuff, sizeof(ssidBuff));
  String hostname(ssidBuff);
  hostname.replace(" ", "-");
  WiFi.hostname(hostname.c_str());
```

```
  if (configStore.getFlag(CONFIG_FLAG_STATIC_IP)) {
    if (!WiFi.config(configStore.staticIP,
                     configStore.staticGW,
                     configStore.staticMask,
                     configStore.staticDNS,
                     configStore.staticDNS2)
    ) {
      DEBUG_PRINT("Failed to configure Static IP");
      config_set_last_error(BLYNK_PROV_ERR_CONFIG);
      BlynkState::set(MODE_ERROR);
      return;
    }
  }

  if (!WiFi.begin(configStore.wifiSSID, configStore.wifiPass)) {
    config_set_last_error(BLYNK_PROV_ERR_CONFIG);
    BlynkState::set(MODE_ERROR);
    return;
  }

  unsigned long timeoutMs = millis() + WIFI_NET_CONNECT_TIMEOUT;
  while ((timeoutMs > millis()) && (WiFi.status() != WL_CONNECTED))
  {
    delay(10);
    app_loop();

    if (!BlynkState::is(MODE_CONNECTING_NET)) {
      WiFi.disconnect();
      return;
    }
  }

  if (WiFi.status() == WL_CONNECTED) {
    IPAddress localip = WiFi.localIP();
    if (configStore.getFlag(CONFIG_FLAG_STATIC_IP)) {
      BLYNK_LOG_IP("Using Static IP: ", localip);
    } else {
      BLYNK_LOG_IP("Using Dynamic IP: ", localip);
    }

    BlynkState::set(MODE_CONNECTING_CLOUD);
  } else {
    config_set_last_error(BLYNK_PROV_ERR_NETWORK);
    BlynkState::set(MODE_ERROR);
  }
}

void enterConnectCloud() {
```

```
  BlynkState::set(MODE_CONNECTING_CLOUD);

  Blynk.config(configStore.cloudToken, configStore.cloudHost, config-
Store.cloudPort);
  Blynk.connect(0);

  unsigned long timeoutMs = millis() + WIFI_CLOUD_CONNECT_TIMEOUT;
  while ((timeoutMs > millis()) &&
         (!Blynk.isTokenInvalid()) &&
         (Blynk.connected() == false))
  {
    delay(10);
    Blynk.run();
    app_loop();
    if (!BlynkState::is(MODE_CONNECTING_CLOUD)) {
      Blynk.disconnect();
      return;
    }
  }

  if (millis() > timeoutMs) {
    DEBUG_PRINT("Timeout");
  }

  if (Blynk.isTokenInvalid()) {
    config_set_last_error(BLYNK_PROV_ERR_TOKEN);
    BlynkState::set(MODE_WAIT_CONFIG);
  } else if (Blynk.connected()) {
    BlynkState::set(MODE_RUNNING);

    if (!configStore.getFlag(CONFIG_FLAG_VALID)) {
      configStore.last_error = BLYNK_PROV_ERR_NONE;
      configStore.setFlag(CONFIG_FLAG_VALID, true);
      config_save();
    }
  } else {
    config_set_last_error(BLYNK_PROV_ERR_CLOUD);
    BlynkState::set(MODE_ERROR);
  }
}

void enterSwitchToSTA() {
  BlynkState::set(MODE_SWITCH_TO_STA);

  DEBUG_PRINT("Switching to STA...");

  delay(1000);
  WiFi.mode(WIFI_OFF);
  delay(100);
```

```
  WiFi.mode(WIFI_STA);

  BlynkState::set(MODE_CONNECTING_NET);
}

void enterError() {
  BlynkState::set(MODE_ERROR);

  unsigned long timeoutMs = millis() + 10000;
  while (timeoutMs > millis() || g_buttonPressed)
  {
    delay(10);
    app_loop();
    if (!BlynkState::is(MODE_ERROR)) {
      return;
    }
  }
  DEBUG_PRINT("Restarting after error.");
  delay(10);

  restartMCU();
}
```

ConfigStore.h

```
#define CONFIG_FLAG_VALID       0x01
#define CONFIG_FLAG_STATIC_IP   0x02

#define BLYNK_PROV_ERR_NONE     0      // All good
#define BLYNK_PROV_ERR_CONFIG   700    // Invalid config from app (malformed to-
ken,etc)
#define BLYNK_PROV_ERR_NETWORK  701    // Could not connect to the router
#define BLYNK_PROV_ERR_CLOUD    702    // Could not connect to the cloud
#define BLYNK_PROV_ERR_TOKEN    703    // Invalid token error (after connection)
#define BLYNK_PROV_ERR_INTERNAL 704    // Other issues (i.e. hardware failure)

struct ConfigStore {
  uint32_t  magic;
  char      version[15];
  uint8_t   flags;

  char      wifiSSID[34];
  char      wifiPass[64];

  char      cloudToken[34];
```

```
  char      cloudHost[34];
  uint16_t  cloudPort;

  uint32_t  staticIP;
  uint32_t  staticMask;
  uint32_t  staticGW;
  uint32_t  staticDNS;
  uint32_t  staticDNS2;

  int       last_error;

  void setFlag(uint8_t mask, bool value) {
    if (value) {
      flags |= mask;
    } else {
      flags &= ~mask;
    }
  }

  bool getFlag(uint8_t mask) {
    return (flags & mask) == mask;
  }
} __attribute__((packed));

ConfigStore configStore;

const ConfigStore configDefault = {
  0x626C6E6B,
  BLYNK_FIRMWARE_VERSION,
  0x00,

  "",
  "",

  "invalid token",
  CONFIG_DEFAULT_SERVER,
  CONFIG_DEFAULT_PORT,
  0,
  BLYNK_PROV_ERR_NONE
};

template<typename T, int size>
void CopyString(const String& s, T(&arr)[size]) {
  s.toCharArray(arr, size);
}

static bool config_load_blnkopt()
{
  static const char blnkopt[] = "blnkopt\0"
```

```
    BLYNK_PARAM_KV("ssid" , BLYNK_PARAM_PLACEHOLDER_64
                            BLYNK_PARAM_PLACEHOLDER_64
                            BLYNK_PARAM_PLACEHOLDER_64
                            BLYNK_PARAM_PLACEHOLDER_64)
    BLYNK_PARAM_KV("host" , CONFIG_DEFAULT_SERVER)
    BLYNK_PARAM_KV("port" , BLYNK_TOSTRING(CONFIG_DEFAULT_PORT))
    "\0";

  BlynkParam prov(blnkopt+8, sizeof(blnkopt)-8-2);
  BlynkParam::iterator ssid = prov["ssid"];
  BlynkParam::iterator pass = prov["pass"];
  BlynkParam::iterator auth = prov["auth"];
  BlynkParam::iterator host = prov["host"];
  BlynkParam::iterator port = prov["port"];

  if (!(ssid.isValid() && auth.isValid())) {
    return false;
  }

  // reset to defaut before loading values from blnkopt
  configStore = configDefault;

  if (ssid.isValid()) { CopyString(ssid.asStr(), configStore.wifiSSID); }
  if (pass.isValid()) { CopyString(pass.asStr(), configStore.wifiPass); }
  if (auth.isValid()) { CopyString(auth.asStr(), configStore.cloudToken); }
  if (host.isValid()) { CopyString(host.asStr(), configStore.cloudHost); }
  if (port.isValid()) { configStore.cloudPort = port.asInt(); }

  return true;
}

#include <EEPROM.h>
#define EEPROM_CONFIG_START 0

void config_load()
{
  memset(&configStore, 0, sizeof(configStore));
  EEPROM.get(EEPROM_CONFIG_START, configStore);
  if (configStore.magic != configDefault.magic) {
    DEBUG_PRINT("Using default config.");
    configStore = configDefault;
    return;
  }
}

bool config_save()
{
  EEPROM.put(EEPROM_CONFIG_START, configStore);
  EEPROM.commit();
```

```
  DEBUG_PRINT("Configuration stored to flash");
  return true;
}

bool config_init()
{
  EEPROM.begin(sizeof(ConfigStore));
  config_load();
  return true;
}

void enterResetConfig()
{
  DEBUG_PRINT("Resetting configuration!");
  configStore = configDefault;
  config_save();
  BlynkState::set(MODE_WAIT_CONFIG);
}

void config_set_last_error(int error) {
  // Only set error if not provisioned
  if (!configStore.getFlag(CONFIG_FLAG_VALID)) {
    configStore = configDefault;
    configStore.last_error = error;
    BLYNK_LOG2("Last error code: ", error);
    config_save();
  }
}
```

Console.h

```
#include <Blynk/BlynkConsole.h>

BlynkConsole    edgentConsole;

void console_init()
{
  edgentConsole.init(BLYNK_PRINT);

  edgentConsole.print("\n>");

  edgentConsole.addCommand("reboot", []() {
    edgentConsole.print(R"json({"status":"OK","msg":"resetting device"})json"
"\n");
    delay(100);
```

```
    restartMCU();
  });

  edgentConsole.addCommand("config", []() {
    edgentConsole.print(R"json({"status":"OK","msg":"entering configuration
mode"})json" "\n");
    BlynkState::set(MODE_WAIT_CONFIG);
  });

  edgentConsole.addCommand("devinfo", []() {
    edgentConsole.printf(
        R"json({"board":"%s","tmpl_id":"%s","fw_type":"%s","fw_ver":"%s"})json"
"\n",
        BLYNK_DEVICE_NAME,
        BLYNK_TEMPLATE_ID,
        BLYNK_FIRMWARE_TYPE,
        BLYNK_FIRMWARE_VERSION
    );
  });

  edgentConsole.addCommand("netinfo", []() {
    char ssidBuff[64];
    getWiFiName(ssidBuff, sizeof(ssidBuff));

    edgentConsole.printf(
        R"json({"ssid":"%s","bssid":"%s","mac":"%s","rssi":%d})json" "\n",
        ssidBuff,
        WiFi.softAPmacAddress().c_str(),
        WiFi.macAddress().c_str(),
        WiFi.RSSI()
    );
  });

}

BLYNK_WRITE(InternalPinDBG) {
  String cmd = String(param.asStr()) + "\n";
  edgentConsole.runCommand((char*)cmd.c_str());
}
```

Edgent_ESP8266__stair_case_robot.ino

```
#define BLYNK_TEMPLATE_ID "TMPL3MU1Qq9cL"
#define BLYNK_DEVICE_NAME "staircaserobot"

#define BLYNK_FIRMWARE_VERSION        "0.1.0"
```

```
#define BLYNK_PRINT Serial

#define APP_DEBUG

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_SPARKFUN_BLYNK_BOARD
#define USE_NODE_MCU_BOARD
//#define USE_WITTY_CLOUD_BOARD
//#define USE_WEMOS_D1_MINI

#include "BlynkEdgent.h"

int forwards=D0,right=D1,left=D2,reverses=D5;
int forwards1=D6,right1=D7,left1=D8,reverses1=3;
int forwardState;
int reversesState;
int leftState;
int rightState;

BLYNK_WRITE(V0) {
  forwardState = param.asInt();
  digitalWrite(forwards, forwardState);
}
BLYNK_WRITE(V1) {
  reversesState = param.asInt();
  digitalWrite(reverses, reversesState);
}
BLYNK_WRITE(V2) {
  leftState = param.asInt();
  digitalWrite(left, leftState);
}
BLYNK_WRITE(V3) {
  rightState = param.asInt();
  digitalWrite(right, rightState);
}

void setup()
{
  Serial.begin(115200);
   pinMode(forwards, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(left, OUTPUT);
  pinMode(reverses, OUTPUT);
  pinMode(forwards1, OUTPUT);
  pinMode(right1, OUTPUT);
  pinMode(left1, OUTPUT);
  pinMode(reverses1, OUTPUT);
  delay(1000);

  BlynkEdgent.begin();
}

void loop() {
  BlynkEdgent.run();
    if(forwardState==HIGH)
{
  digitalWrite(forwards,HIGH);
digitalWrite(right,LOW);
digitalWrite(left,LOW);
digitalWrite(reverses,HIGH);
```

```
 digitalWrite(forwards1,HIGH);
digitalWrite(right1,LOW);
digitalWrite(left1,LOW);
digitalWrite(reverses1,HIGH);

Serial.println("forward");
}
 else if(reversesState==HIGH)
{
digitalWrite(forwards,LOW);
digitalWrite(right,HIGH);
digitalWrite(left,HIGH);
digitalWrite(reverses,LOW);

digitalWrite(forwards1,LOW);
digitalWrite(right1,HIGH);
digitalWrite(left1,HIGH);
digitalWrite(reverses1,LOW);
Serial.println("REVERSE");
}
else if(leftState==HIGH)
{
digitalWrite(forwards,LOW);
digitalWrite(right,HIGH);
digitalWrite(left,LOW);
digitalWrite(reverses,HIGH);

digitalWrite(forwards1,LOW);
digitalWrite(right1,HIGH);
digitalWrite(left1,LOW);
digitalWrite(reverses1,HIGH);
Serial.println("LEFT");
}
 else if(rightState==HIGH)
{
digitalWrite(forwards,HIGH);
digitalWrite(right,LOW);
digitalWrite(left,HIGH);
digitalWrite(reverses,LOW);

digitalWrite(forwards1,HIGH);
digitalWrite(right1,LOW);
digitalWrite(left1,HIGH);
digitalWrite(reverses1,LOW);
Serial.println("RIGHT");
}
else
{
 digitalWrite(forwards,LOW);
digitalWrite(right,LOW);
digitalWrite(left,LOW);
digitalWrite(reverses,LOW);

digitalWrite(forwards1,LOW);
digitalWrite(right1,LOW);
digitalWrite(left1,LOW);
digitalWrite(reverses1,LOW);
Serial.println("STOP");
}
```

```
delay(50);
}
```

Indicator.h

```
#if defined(BOARD_LED_PIN_WS2812)
  #include <Adafruit_NeoPixel.h>    // Library:
https://github.com/adafruit/Adafruit_NeoPixel

  Adafruit_NeoPixel rgb = Adafruit_NeoPixel(1, BOARD_LED_PIN_WS2812, NEO_GRB +
NEO_KHZ800);
#endif

void indicator_run();

#if !defined(BOARD_LED_BRIGHTNESS)
#define BOARD_LED_BRIGHTNESS 255
#endif

#if defined(BOARD_LED_PIN_WS2812) || defined(BOARD_LED_PIN_R)
#define BOARD_LED_IS_RGB
#endif

#define DIMM(x)    ((uint32_t)(x)*(BOARD_LED_BRIGHTNESS)/255)
#define RGB(r,g,b) (DIMM(r) << 16 | DIMM(g) << 8 | DIMM(b) << 0)
#define TO_PWM(x)  ((uint32_t)(x)*(BOARD_PWM_MAX)/255)

class Indicator {
public:

  enum Colors {
    COLOR_BLACK   = RGB(0x00, 0x00, 0x00),
    COLOR_WHITE   = RGB(0xFF, 0xFF, 0xE7),
    COLOR_BLUE    = RGB(0x0D, 0x36, 0xFF),
    COLOR_BLYNK   = RGB(0x2E, 0xFF, 0xB9),
    COLOR_RED     = RGB(0xFF, 0x10, 0x08),
    COLOR_MAGENTA = RGB(0xA7, 0x00, 0xFF),
  };

  Indicator() {
  }

  void init() {
    m_Counter = 0;
    initLED();
  }
```

```
  uint32_t run() {
    State currState = BlynkState::get();

    // Reset counter if indicator state changes
    if (m_PrevState != currState) {
      m_PrevState = currState;
      m_Counter = 0;
    }

    const long t = millis();
    if (g_buttonPressed) {
      if (t - g_buttonPressTime > BUTTON_HOLD_TIME_ACTION)     { return beat-
LED(COLOR_WHITE,   (int[]){ 100, 100 }); }
      if (t - g_buttonPressTime > BUTTON_HOLD_TIME_INDICATION) { return
waveLED(COLOR_WHITE,   1000); }
    }
    switch (currState) {
    case MODE_RESET_CONFIG:
    case MODE_WAIT_CONFIG:       return beatLED(COLOR_BLUE,    (int[]){ 50, 500 });
    case MODE_CONFIGURING:       return beatLED(COLOR_BLUE,    (int[]){ 200, 200
});
    case MODE_CONNECTING_NET:    return beatLED(COLOR_BLYNK,   (int[]){ 50, 500 });
    case MODE_CONNECTING_CLOUD:  return beatLED(COLOR_BLYNK,   (int[]){ 100, 100
});
    case MODE_RUNNING:           return waveLED(COLOR_BLYNK,   5000);
    case MODE_OTA_UPGRADE:       return beatLED(COLOR_MAGENTA, (int[]){ 50, 50 });
    default:                     return beatLED(COLOR_RED,     (int[]){ 80, 100,
80, 1000 } );
    }
  }

protected:

  /*
   * LED drivers
   */

#if defined(BOARD_LED_PIN_WS2812)  // Addressable, NeoPixel RGB LED

  void initLED() {
    rgb.begin();
    setRGB(COLOR_BLACK);
  }

  void setRGB(uint32_t color) {
    rgb.setPixelColor(0, color);
    rgb.show();
  }
```

```
#elif defined(BOARD_LED_PIN_R)     // Normal RGB LED (common anode or common cath-
ode)

  void initLED() {
    pinMode(BOARD_LED_PIN_R, OUTPUT);
    pinMode(BOARD_LED_PIN_G, OUTPUT);
    pinMode(BOARD_LED_PIN_B, OUTPUT);
  }

  void setRGB(uint32_t color) {
    uint8_t r = (color & 0xFF0000) >> 16;
    uint8_t g = (color & 0x00FF00) >> 8;
    uint8_t b = (color & 0x0000FF);
    #if BOARD_LED_INVERSE
    analogWrite(BOARD_LED_PIN_R, TO_PWM(255 - r));
    analogWrite(BOARD_LED_PIN_G, TO_PWM(255 - g));
    analogWrite(BOARD_LED_PIN_B, TO_PWM(255 - b));
    #else
    analogWrite(BOARD_LED_PIN_R, TO_PWM(r));
    analogWrite(BOARD_LED_PIN_G, TO_PWM(g));
    analogWrite(BOARD_LED_PIN_B, TO_PWM(b));
    #endif
  }

#elif defined(BOARD_LED_PIN)       // Single color LED

  void initLED() {
    pinMode(BOARD_LED_PIN, OUTPUT);
  }

  void setLED(uint32_t color) {
    #if BOARD_LED_INVERSE
    analogWrite(BOARD_LED_PIN, TO_PWM(255 - color));
    #else
    analogWrite(BOARD_LED_PIN, TO_PWM(color));
    #endif
  }

#else

  #warning Invalid LED configuration.

  void initLED() {
  }

  void setLED(uint32_t color) {
  }

#endif
```

```
  /*
   * Animations
   */

  uint32_t skipLED() {
    return 20;
  }

#if defined(BOARD_LED_IS_RGB)

  template<typename T>
  uint32_t beatLED(uint32_t onColor, const T& beat) {
    const uint8_t cnt = sizeof(beat)/sizeof(beat[0]);
    setRGB((m_Counter % 2 == 0) ? onColor : (uint32_t)COLOR_BLACK);
    uint32_t next = beat[m_Counter % cnt];
    m_Counter = (m_Counter+1) % cnt;
    return next;
  }

  uint32_t waveLED(uint32_t colorMax, unsigned breathePeriod) {
    uint8_t redMax = (colorMax & 0xFF0000) >> 16;
    uint8_t greenMax = (colorMax & 0x00FF00) >> 8;
    uint8_t blueMax = (colorMax & 0x0000FF);

    // Brightness will rise from 0 to 128, then fall back to 0
    uint8_t brightness = (m_Counter < 128) ? m_Counter : 255 - m_Counter;

    // Multiply our three colors by the brightness:
    redMax *= ((float)brightness / 128.0);
    greenMax *= ((float)brightness / 128.0);
    blueMax *= ((float)brightness / 128.0);
    // And turn the LED to that color:
    setRGB((redMax << 16) | (greenMax << 8) | blueMax);

    // This function relies on the 8-bit, unsigned m_Counter rolling over.
    m_Counter = (m_Counter+1) % 256;
    return breathePeriod / 256;
  }

#else

  template<typename T>
  uint32_t beatLED(uint32_t, const T& beat) {
    const uint8_t cnt = sizeof(beat)/sizeof(beat[0]);
    setLED((m_Counter % 2 == 0) ? BOARD_LED_BRIGHTNESS : 0);
    uint32_t next = beat[m_Counter % cnt];
    m_Counter = (m_Counter+1) % cnt;
    return next;
```

```
  }

  uint32_t waveLED(uint32_t, unsigned breathePeriod) {
    uint32_t brightness = (m_Counter < 128) ? m_Counter : 255 - m_Counter;

    setLED(DIMM(brightness*2));

    // This function relies on the 8-bit, unsigned m_Counter rolling over.
    m_Counter = (m_Counter+1) % 256;
    return breathePeriod / 256;
  }

#endif

private:
  uint8_t m_Counter;
  State   m_PrevState;
};

Indicator indicator;

/*
 * Animation timers
 */

#if defined(USE_TICKER)

  #include <Ticker.h>

  Ticker blinker;

  void indicator_run() {
    uint32_t returnTime = indicator.run();
    if (returnTime) {
      blinker.attach_ms(returnTime, indicator_run);
    }
  }

  void indicator_init() {
    indicator.init();
    blinker.attach_ms(100, indicator_run);
  }

#elif defined(USE_PTHREAD)

  #include <pthread.h>

  pthread_t blinker;
```

```
  void* indicator_thread(void*) {
    while (true) {
      uint32_t returnTime = indicator.run();
      returnTime = BlynkMathClamp(returnTime, 1, 10000);
      vTaskDelay(returnTime);
    }
  }

  void indicator_init() {
    indicator.init();
    pthread_create(&blinker, NULL, indicator_thread, NULL);
  }

#elif defined(USE_TIMER_ONE)

  #include <TimerOne.h>

  void indicator_run() {
    uint32_t returnTime = indicator.run();
    if (returnTime) {
      Timer1.initialize(returnTime*1000);
    }
  }

  void indicator_init() {
    indicator.init();
    Timer1.initialize(100*1000);
    Timer1.attachInterrupt(indicator_run);
  }

#elif defined(USE_TIMER_THREE)

  #include <TimerThree.h>

  void indicator_run() {
    uint32_t returnTime = indicator.run();
    if (returnTime) {
      Timer3.initialize(returnTime*1000);
    }
  }

  void indicator_init() {
    indicator.init();
    Timer3.initialize(100*1000);
    Timer3.attachInterrupt(indicator_run);
  }

#elif defined(USE_TIMER_FIVE)
```

```
  #include <Timer5.h>    // Library: https://github.com/michael71/Timer5

  int indicator_counter = -1;
  void indicator_run() {
    indicator_counter -= 10;
    if (indicator_counter < 0) {
      indicator_counter = indicator.run();
    }
  }

  void indicator_init() {
    indicator.init();
    MyTimer5.begin(1000/10);
    MyTimer5.attachInterrupt(indicator_run);
    MyTimer5.start();
  }

#else

  #warning LED indicator needs a functional timer!

  void indicator_run() {}
  void indicator_init() {}

#endif
```

OTA.h

```
#define OTA_FATAL(...) { BLYNK_LOG1(__VA_ARGS__); delay(1000); restartMCU(); }

#define USE_SSL

String overTheAirURL;

extern BlynkTimer edgentTimer;

BLYNK_WRITE(InternalPinOTA) {
  overTheAirURL = param.asString();

  edgentTimer.setTimeout(2000L, [](){
    // Start OTA
    Blynk.logEvent("sys_ota", "OTA started");

    // Disconnect, not to interfere with OTA process
    Blynk.disconnect();
```

```
    BlynkState::set(MODE_OTA_UPGRADE);
  });
}

#if defined(ESP32)
  #include <Update.h>
  #include <WiFiClientSecure.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
  #include <WiFiClientSecure.h>
  #include <time.h>
#endif

#if defined(USE_SSL) && defined(ESP8266)

WiFiClient* connectSSL(const String& host, const int port)
{
  WiFiUDP::stopAll();
  WiFiClient::stopAll();

  time_t now = time(nullptr);
  if (time(nullptr) < 100000) {
    // Synchronize time useing SNTP. This is necessary to verify that
    // the TLS certificates offered by the server are currently valid
    configTime(0, 0, "pool.ntp.org", "time.nist.gov");

    while (now < 100000) {
        delay(100);
        now = time(nullptr);
    }
  }

  // Reuse Secure WIFI Client on ESP8266
  //WiFiClientSecure* clientSSL = &_blynkWifiClient;
  WiFiClientSecure* clientSSL = new WiFiClientSecure();

  clientSSL->setTrustAnchors(&BlynkCert);
  if (!clientSSL->connect(host.c_str(), port)) {
    OTA_FATAL(F("Connection failed"));
  }
  return clientSSL;
}

#elif defined(USE_SSL) && defined(ESP32)

WiFiClient* connectSSL(const String& host, const int port)
{
  WiFiUDP::stopAll();
```

```
  WiFiClient::stopAll();

  WiFiClientSecure* clientSSL = new WiFiClientSecure();
  clientSSL->setCACert(BLYNK_DEFAULT_ROOT_CA);
  if (clientSSL->connect(host.c_str(), port)) {
    DEBUG_PRINT(F("Certificate OK"));
  } else {
    OTA_FATAL(F("Secure connection failed"));
  }
  return clientSSL;
}

#endif

WiFiClient* connectTCP(const String& host, const int port)
{
  WiFiUDP::stopAll();
  WiFiClient::stopAll();

  WiFiClient* clientTCP = new WiFiClient();
  if (!clientTCP->connect(host.c_str(), port)) {
    OTA_FATAL(F("Client not connected"));
  }
  return clientTCP;
}

bool parseURL(String url, String& protocol, String& host, int& port, String& uri)
{
  int index = url.indexOf(':');
  if(index < 0) {
    return false;
  }

  protocol = url.substring(0, index);
  url.remove(0, (index + 3)); // remove protocol part

  index = url.indexOf('/');
  String server = url.substring(0, index);
  url.remove(0, index);       // remove server part

  index = server.indexOf(':');
  if(index >= 0) {
    host = server.substring(0, index);          // hostname
    port = server.substring(index + 1).toInt(); // port
  } else {
    host = server;
    if (protocol == "http") {
      port = 80;
    } else if (protocol == "https") {
```

```
      port = 443;
    }
  }

  if (url.length()) {
    uri = url;
  } else {
    uri = "/";
  }
  return true;
}

void enterOTA() {
  BlynkState::set(MODE_OTA_UPGRADE);

  // Disconnect, not to interfere with OTA process
  Blynk.disconnect();

  String protocol, host, url;
  int port;

  DEBUG_PRINT(String("OTA: ") + overTheAirURL);

  if (!parseURL(overTheAirURL, protocol, host, port, url)) {
    OTA_FATAL(F("Cannot parse URL"));
  }

  DEBUG_PRINT(String("Connecting to ") + host + ":" + port);

  Client* client = NULL;
  if (protocol == "http") {
    client = connectTCP(host, port);
#ifdef USE_SSL
  } else if (protocol == "https") {
    client = connectSSL(host, port);
#endif
  } else {
    OTA_FATAL(String("Unsupported protocol: ") + protocol);
  }

  client->print(String("GET ") + url + " HTTP/1.0\r\n"
               + "Host: " + host + "\r\n"
               + "Connection: keep-alive\r\n"
               + "\r\n");

  uint32_t timeout = millis();
  while (client->connected() && !client->available()) {
    if (millis() - timeout > 10000L) {
      OTA_FATAL("Response timeout");
```

```
    }
    delay(10);
  }

  // Collect headers
  String md5;
  int contentLength = 0;

  while (client->available()) {
    String line = client->readStringUntil('\n');
    line.trim();
    //DEBUG_PRINT(line);    // Uncomment this to show response headers
    line.toLowerCase();
    if (line.startsWith("content-length:")) {
      contentLength = line.substring(line.lastIndexOf(':') + 1).toInt();
    } else if (line.startsWith("x-md5:")) {
      md5 = line.substring(line.lastIndexOf(':') + 1);
    } else if (line.length() == 0) {
      break;
    }
    delay(10);
  }

  if (contentLength <= 0) {
    OTA_FATAL("Content-Length not defined");
  }

  bool canBegin = Update.begin(contentLength);
  if (!canBegin) {
    Update.printError(BLYNK_PRINT);
    OTA_FATAL("OTA begin failed");
  }

  if (md5.length()) {
    md5.trim();
    md5.toLowerCase();
    DEBUG_PRINT(String("Expected MD5: ") + md5);
    if(!Update.setMD5(md5.c_str())) {
      OTA_FATAL("Cannot set MD5");
    }
  }

  DEBUG_PRINT("Flashing...");

  // The next loop does approx. the same thing as Update.writeStream(http) or Up-
date.write(http)

  int written = 0;
  int prevProgress = 0;
```

```
  uint8_t buff[256];
  while (client->connected() && written < contentLength) {
    delay(10);
    timeout = millis();
    while (client->connected() && !client->available()) {
      delay(1);
      if (millis() - timeout > 10000L) {
        OTA_FATAL("Timeout");
      }
    }

    int len = client->read(buff, sizeof(buff));
    if (len <= 0) continue;

    Update.write(buff, len);
    written += len;

    const int progress = (written*100)/contentLength;
    if (progress - prevProgress >= 10 || progress == 100) {
      BLYNK_PRINT.print(String("\r ") + progress + "%");
      prevProgress = progress;
    }
  }
  BLYNK_PRINT.println();
  client->stop();

  if (written != contentLength) {
    Update.printError(BLYNK_PRINT);
    OTA_FATAL(String("Write failed. Written ") + written + " / " + contentLength +
" bytes");
  }

  if (!Update.end()) {
    Update.printError(BLYNK_PRINT);
    OTA_FATAL(F("Update not ended"));
  }

  if (!Update.isFinished()) {
    OTA_FATAL(F("Update not finished"));
  }

  DEBUG_PRINT("=== Update successfully completed. Rebooting.");
  restartMCU();
}
```

Reset.h

```
#ifdef BOARD_BUTTON_PIN

volatile bool     g_buttonPressed = false;
volatile uint32_t g_buttonPressTime = -1;

void button_action(void)
{
  BlynkState::set(MODE_RESET_CONFIG);
}

ICACHE_RAM_ATTR
void button_change(void)
{
#if BOARD_BUTTON_ACTIVE_LOW
  bool buttonState = !digitalRead(BOARD_BUTTON_PIN);
#else
  bool buttonState = digitalRead(BOARD_BUTTON_PIN);
#endif

  if (buttonState && !g_buttonPressed) {
    g_buttonPressTime = millis();
    g_buttonPressed = true;
    DEBUG_PRINT("Hold the button for 10 seconds to reset configuration...");
  } else if (!buttonState && g_buttonPressed) {
    g_buttonPressed = false;
    uint32_t buttonHoldTime = millis() - g_buttonPressTime;
    if (buttonHoldTime >= BUTTON_HOLD_TIME_ACTION) {
      button_action();
    } else if (buttonHoldTime >= BUTTON_PRESS_TIME_ACTION) {
      // User action
    }
    g_buttonPressTime = -1;
  }
}

void button_init()
{
#if BOARD_BUTTON_ACTIVE_LOW
  pinMode(BOARD_BUTTON_PIN, INPUT_PULLUP);
#else
  pinMode(BOARD_BUTTON_PIN, INPUT);
#endif
  attachInterrupt(BOARD_BUTTON_PIN, button_change, CHANGE);
}
```

```
#else

#define g_buttonPressed     false
#define g_buttonPressTime   0

void button_init() {}

#endif
```

Settings.h

```
/*
 * Board configuration (see examples below).
 */

#if defined(USE_NODE_MCU_BOARD) || defined(USE_WEMOS_D1_MINI)

  #define BOARD_BUTTON_PIN            0
  #define BOARD_BUTTON_ACTIVE_LOW     true

  #define BOARD_LED_PIN               2
  #define BOARD_LED_INVERSE           true
  #define BOARD_LED_BRIGHTNESS        255

#elif defined(USE_SPARKFUN_BLYNK_BOARD)

  #define BOARD_BUTTON_PIN            0
  #define BOARD_BUTTON_ACTIVE_LOW     true

  #define BOARD_LED_PIN_WS2812        4
  #define BOARD_LED_BRIGHTNESS        64

#elif defined(USE_WITTY_CLOUD_BOARD)

  #define BOARD_BUTTON_PIN            4
  #define BOARD_BUTTON_ACTIVE_LOW     true

  #define BOARD_LED_PIN_R             15
  #define BOARD_LED_PIN_G             12
  #define BOARD_LED_PIN_B             13
  #define BOARD_LED_INVERSE           false
  #define BOARD_LED_BRIGHTNESS        64

#else
```

```
  #warning "Custom board configuration is used"

  #define BOARD_BUTTON_PIN            0                   // Pin where user but-
ton is attached
  #define BOARD_BUTTON_ACTIVE_LOW     true                // true if button is
"active-low"

  #define BOARD_LED_PIN               4                   // Set LED pin - if you
have a single-color LED attached
  //#define BOARD_LED_PIN_R           15                  // Set R,G,B pins - if
your LED is PWM RGB
  //#define BOARD_LED_PIN_G           12
  //#define BOARD_LED_PIN_B           13
  //#define BOARD_LED_PIN_WS2812      4                   // Set if your LED is
WS2812 RGB
  #define BOARD_LED_INVERSE           false               // true if LED is com-
mon anode, false if common cathode
  #define BOARD_LED_BRIGHTNESS        64                  // 0..255 brightness
control

#endif


/*
 * Advanced options
 */

#define BUTTON_HOLD_TIME_INDICATION   3000
#define BUTTON_HOLD_TIME_ACTION       10000
#define BUTTON_PRESS_TIME_ACTION      50

#define BOARD_PWM_MAX                 1023

#define CONFIG_AP_URL                 "blynk.setup"
#define CONFIG_DEFAULT_SERVER         "blynk.cloud"
#define CONFIG_DEFAULT_PORT           443

#define WIFI_NET_CONNECT_TIMEOUT      50000
#define WIFI_CLOUD_CONNECT_TIMEOUT    50000
#define WIFI_AP_IP                    IPAddress(192, 168, 4, 1)
#define WIFI_AP_Subnet                IPAddress(255, 255, 255, 0)
//#define WIFI_CAPTIVE_PORTAL_ENABLE

#define USE_TICKER
//#define USE_TIMER_ONE
//#define USE_TIMER_THREE
//#define USE_TIMER_FIVE
//#define USE_PTHREAD
```

```
#define BLYNK_NO_DEFAULT_BANNER

#if defined(APP_DEBUG)
  #define DEBUG_PRINT(...) BLYNK_LOG1(__VA_ARGS__)
#else
  #define DEBUG_PRINT(...)
#endif
```

5.2 ANDROID

Android is a mobile operating system based on a modified version of the linux kernal and other open source software, designed primarily for touchscreen mobile devices such as smartphone and tablets. Android is developed by a consortium of developers known as the open handset alliance, with the main contributor and commercial marketer being google. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019. The core Android source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache license. This has allowed variants of Android to be developed on a range of other electronics, such as game console, digital cameras, Pc and others, each with a specialized user interface. Some well known derivatives include Android TVfor televisions and wear os for wearables, both developed by Google.

5.2.2 MIT APP INVENTOR

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software (apps) for two operating systems (OS): Android (operating system)|Android, and iOS, which, as of 8 July 2019, is in final beta testing. It is free and open-source software released under Multi-licensing|dual licensing: a Creative Commons Attribution ShareAlike 3.0 Unported license, and an Apache License 2.0 for the source code.

It uses a graphical user interface (GUI) very similar to the programming languages Scratch (programming language) and the StarLogo, which allows users to drag and drop visual objects to create an application that can run on mobile devices. In creating App Inventor, Google drew upon significant prior research in educational computing, and work done within Google on online development environments.

App Inventor and the projects on which it is based are informed by constructionist learning theories, which emphasize that programming can be a vehicle for engaging powerful ideas through active learning.

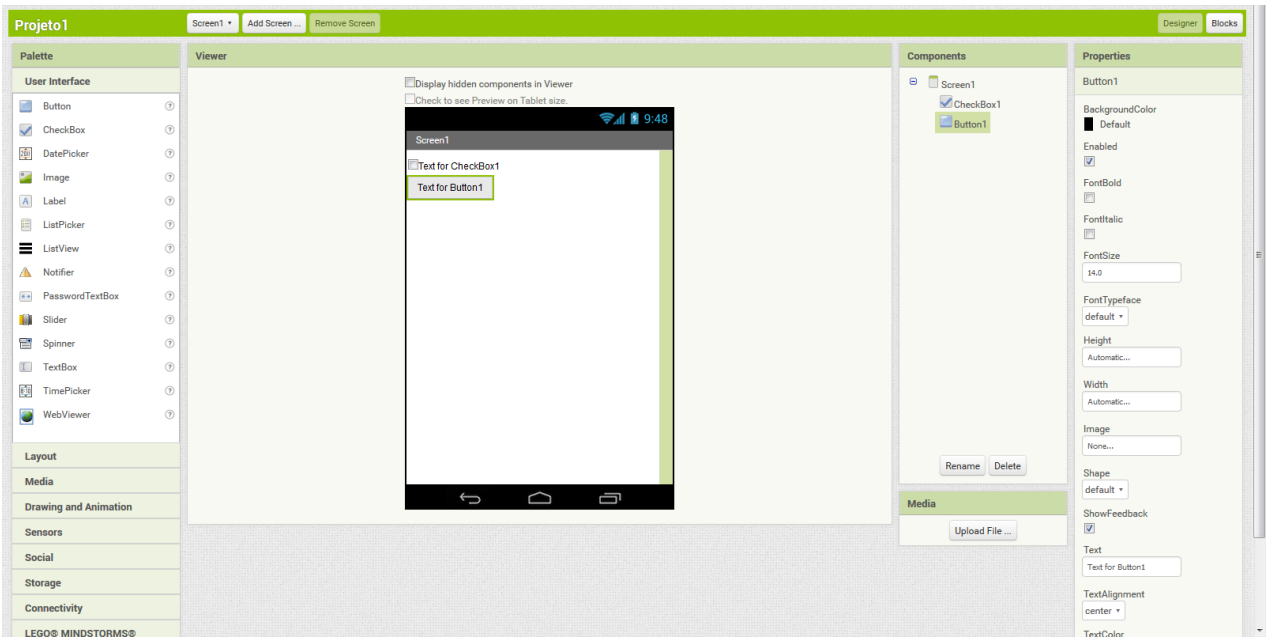


FIG 5.2 App inventor development window

In this project we have made use of App inventor to develop a Lawnmower control app. The app can be connected to the wheel chair through Bluetooth. The app has a user interfacing with buttons for forward, reverse, left and right movement of the Lawnmower.

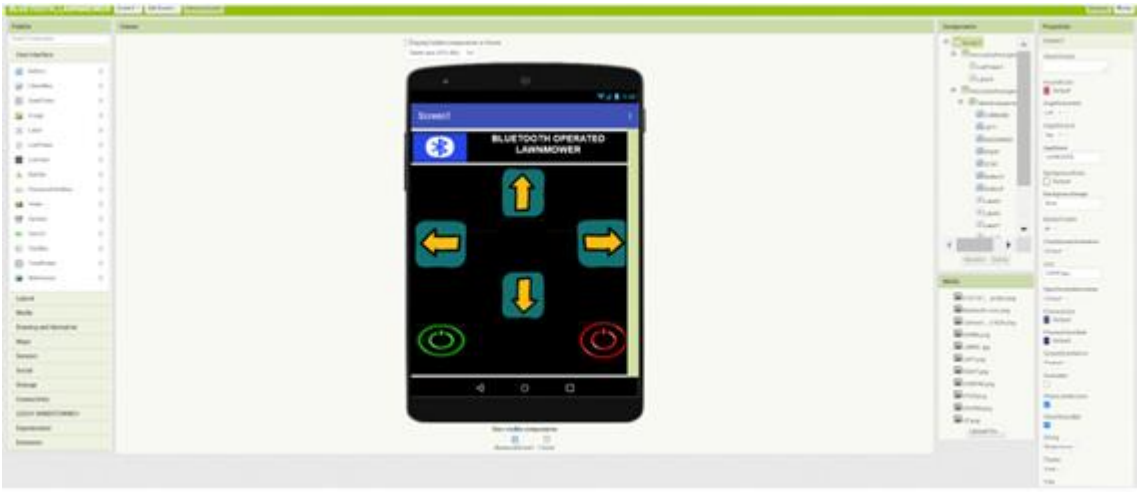


FIG 5.3 Front end of the Lawnmower control app

Our App's front is designed with buttons for all four directions and blade operations. The user can use this buttons to operate the Stair climbing robot. When a button is clicked a command transmitted via the smarts phone Bluetooth to the lawnmowers Bluetooth this command is then received by Arduino and it performs the desired action. We have provided provision to connect to the Bluetooth health monitoring device. The back end programming was done using block programming language. Here we have developed an algorithm with options for Bluetooth connection and data manipulation.

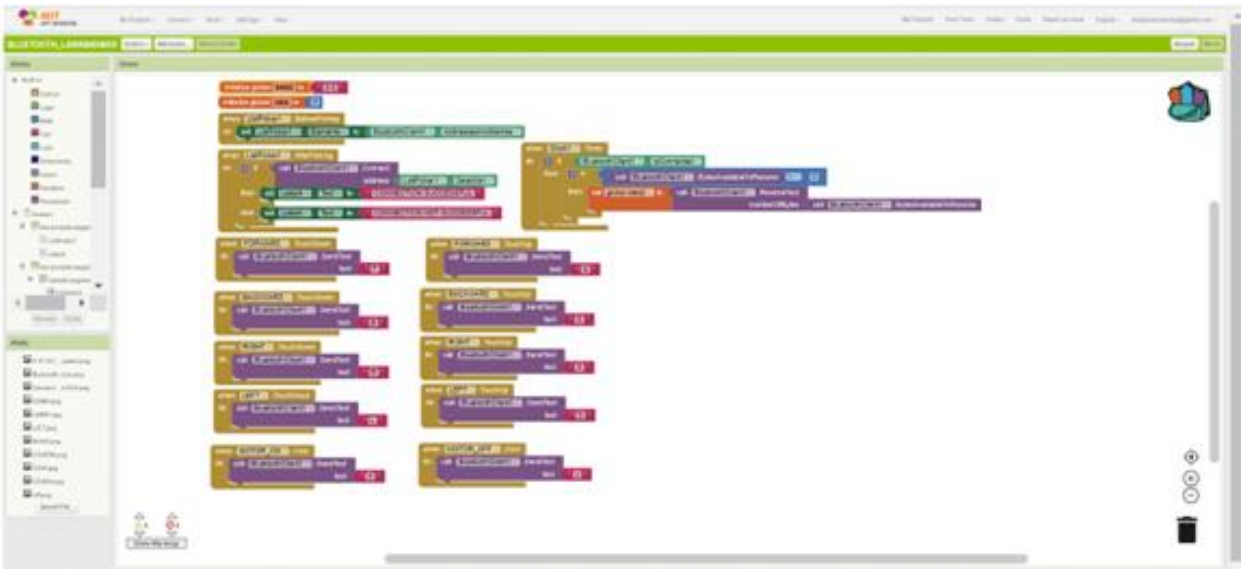


FIG5.4Back end of the Lawnmower control app

5.2.3 APP INVENTOR CLASSIC BLOCKS EDITOR

The blocks editor in the original version ran in a separate Java process, using the Open Blocks Java library for creating visual blocks programming languages and programming. Open Blocks is distributed by MIT's Scheller Teacher Education Program (STEP) and is derived from master's thesis research by Ricarose Roque. Professor Eric Klopfer and Daniel Wendel of the Scheller Program supported the distribution of Open Blocks under an MIT License.Open Blocks visual programming is closely related to StarLogo TNG, a project of STEP, and Scratch, a project of the MIT Media Lab's Lifelong Kindergarten Group led by Mitchelresnick. App Inventor 2 replaced Open Blocks with , Block a blocks editor that runs within a web b

CHAPTER 6

CONCLUSION

This work shows how six-wheel robot works on different surfaces. Robots can climb the stair with great stability. Robot is able to climb steps of height of about 6 inches. Due to hardware limitations such as weight of the frame and design of Staircase Climbing Robot wheel, the robot is able to climb a step of height limited to maximum of 6 inches. This is advantage over geared and leg based robots which have less productive due to poor mobility in a rough terrain. The movement of this robot in controlled using Bluetooth using a smartphone .The robot is able to turn its direction of movement and can maneuver in all directions.

REFERENCES

1. Moh Myint Maung, Soe Nay Lynn Aung , ”Stair Climbing Robot with Stable Platform, “International Journal of Trend in Scientific Research and Development (IJTSRD) Volume: 3 | Issue: 4 | May-Jun 2019 .
2. Akash Asalekar, Akash Chorage, Bhushan Jagdale, Rohit Kusalkar, Shantanu Garud, Vipul Gaikwad, “Design and Fabrication of Staircase Climbing Robot ”International Research Journal of Engineering and Technology (IRJET),Volume: 04 Issue: 07 | July -2017.
3. Tumula Mani Kota Rajasekhar, M.Sugadev, “Arduino Controlled Special Stair Climbing Wheel Chair Bot”, International Journal of Pure and Applied Mathematics Volume 118 No. 24,2018.
4. Ramachandran N, Neelesh kumar N , Rajeshwaran S K , Jeyaseelan R, Sree Harrish S Design And Fabrication Of Semi-Automated lawn mower”, International Journal of Innovative Technology and Exploring Engineering, Volume-8 Issue-10, August 2019.
5. Dipin.A 1 and Dr. Chandrasekhar.T.K, “Solar Powered Vision Based Robotic lawnmower“International Journal of Engineering Research and Reviews, Vol. 2, Issue 2, pp: 53-56, June 2014.
6. Puneet Kaushik , Mohit Jain , Gayatri Patidar ,Paradayil Rhea Eapen , Chandra Prabha Sharma, “Smart Floor Cleaning Robot Using Android”, International Journal of Electronics Engineering , Volume 10, Issue 2, pp. 502-506, Dec 2018.
7. K Aruna Manjusha, S V S Prasad, B.Naresh, S Monika, “Design and Implementation of Smart Floor Cleaning Robot using Android App”,

International Journal of Innovative Technology and Exploring Engineering, Volume-8, Issue-4, March 2019.

8. Manya Jain, Pankaj Singh Rawat, Jyoti Morbale, “Automatic Floor Cleaner”, International Research Journal of Engineering and Technology, Vol: 4, No: 4, Apr -2017.

9. Hala Jamal, Dr. Salih Al-Qaraawi, “Obstacle Avoidance Based on Ultrasonic Sensors and Optical Encoders “, International Journal of Advanced Research in Computer Engineering & Technology, Volume 8, Issue 8, August 2019.

10. Pavithra A C, Subramanya Goutham V, “Obstacle Avoidance Robot Using Arduino”, International Journal of Engineering Research & Technology, Volume 6, Issue 13, 2018.