TITLE OF THE PROJECT

A PROJECT REPORT

Submitted by,

PAPPUVSSSUBRAHMANYESWARA REDDY - 20201CEI0144

NAGIREDDYGARI JASWANTHKUMAR REDDY -20201CEI0097

MITTAPALLI SIVA SAI SANTHOSH -20201CEI0124

Under the guidance of,

Ms. Shilpa C N

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN

COMPUTER ENGINEERING [ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING]

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **“Advance traffic navigation system(using IOT)”** being submitted by “Pappu V S S Subrahmanyeswara Reddy (20201CEI0144) , Nagireddy Gari Jaswanth Kumar Reddy (20201CEI0097) ,Mittapalli Siva Sai Santhosh (20201CEI0124)”in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Engineering[Artificial Intelligence and Machine Learning] is a bonafide work carried out under my supervision.

Ms. Shilpa C N	**Dr.Gopal K. Shyam**
Assistant professor Grade-1	Prof.&HoD
School of CSE	School of CSE
Presidency University	Presidency University

Dr. C. KALAIARASAN	**Dr. L. SHAKKEERA**	**Dr. Md. SAMEERUDDIN KHAN**
Associate Dean	Associate Dean	Dean
School of CSE&IS	School of CSE&IS	School of CSE&IS
Presidency University	Presidency University	Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **“Advance Navigation Traffic System”**in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Engineering(Artificial Intelligence & Machine Learning)**, is a record of our own investigations carried under the guidance of **Shilpa C N, Assistant professor Grade-1,School of Computer Science and Engineering, Presidency University, Bengaluru.**

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

Students Names	**Roll Numbers**	**Signature’s**
Pappu V S SSubrahmanyeswara Reddy	20201CEI0144	
Nagireddy Gari Jaswanth Kumar Reddy	20201CEI0097	
Mittapalli Siva Sai Santhosh	20201CEI0124	

ABSTRACT

India is the world's second largest populated country and a fast growing economy, The country have the primary mode of transportation system , Indian roads handle nearly 88 percent of the country's passenger traffic and 12 percent of its freight. However, the many of Indian roads have an issues like potholes , humps , congestion, poor surface quality, and insufficient maintenance.

There is a common feature on these roads is the presence of speed breakers designed to control vehicle speed and prevent accidents. Unfortunately, the distribution of these speed breakers is uneven, and their heights often lack a scientific basis. Additionally, the formation of potholes, exacerbated by heavy rains and the movement of large vehicles, poses a significant risk .It leading to danger accidents and loss of lives in India countries.

These challenges, there is a pressing need for a cost-effective solution that not only gathers information about the Number of potholes and speed breakers but also advance information to drivers in navigating safely. The proposed system aims to empower drivers to proactively avoid accidents caused by potholes and raised humps, There by enhancing road safety.

Keywords : 4 - Dc motors, 4-wheels ,Pothole-detector, Hump-detector, Wifi - module esp32

Gsm-motherboard , 3-Batteries, Power supply, GPS Locator, drivers board.

ACKNOWLEDGEMENT

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

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Scienceand Engineering&School of Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science and Engineering & School of Information Science, Presidency University and Dr. Gopal K. Shyam, Head of the Department, School of Computer Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Ms. SHILPA C N, Assistant professor Grade-1**, School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project CoordinatorsDr. Sasidhar Babu S., Dr. Sudha P., and Ms. Yogeetha B. R.
We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Pappu V S SSubrahmanyeswara Reddy (20201CEI0144)
Nagireddy Jaswanth Kumar Reddy (20201CEI0097)
Mittapalli Siva Sai Santhosh (20201CEI0124)

LIST OF TABLES

Sl. No.	**Table Name**	**Table Caption**	**Page No.**
1	Table 6.1	PORT Function Multiplexing	31

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 4.1	METHODOLOGY	10
2	Figure 6.1	Architecture of cloud	14
3	Figure 6.2	Blynk Architecture	16
4	Figure 6.2.1	Blynk Architecture	17
5	Figure 6.2.2	Blynk Architecture	18
6	Figure 6.2.3	Blynk Architecture	19
7	Figure 6.2.4	Blynk Architecture	20
8	Figure 6.2.5	Blynk Architecture	21
9	Figure 6.2.6	Blynk Architecture	22
10	Figure 6.2.7	Blynk Architecture	23
11	Figure 6.2.8	Blynk Architecture	23
12	Figure 6.3.1	Blynk Architecture	27
13	Figure 6.3.2	Arduino uno	29
14	Figure 7.1	Time line of project	35

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iv**
	ACKNOWLEDGMENT	**v**
	…	**…**
1.	**INTRODUCTION**	04
	1.1 Definitions,and Acronyms	05
	1.1.1 IoT (Internet of Things)	05
	1.1.2 Cloud Computing	05
	1.1.3 Arduino	05
	1.1.4 Blynk App	05
2.	**LITERTURE SURVEY**	06
3.	**RESEARCH GAPS OF EXISTING METHODS**	**08**
	3.1Accuracy Improvement	08
	3.2Real-time Data Processing	08
	3.3Adaptability to Road Conditions	08
	3.4Cost-effective Sensor Integration	08
	3.5Integration with Autonomous Vehicles:	08
	3.6User-Friendly Notifications	08
	3.7Network Security and Reliability	09
	3.8Long-Term System Maintenance	09
4.	**PROPOSED METHODOLOGY**	**10**
5	**OBJECTIVES**	**12**
6	**SYSTEM DESIGN & IMPLEMENTATION**	**13**
	6.1 ARDUINO	13
	6.2 BLYNK APP	15
	6.3 Hardware	27
	6.4 Digital Pins	33

7	**TIMELINE FOR EXECUTION OF PROJECT**	**35**
8	**OUTCOMES**	**36**
	8.1Real-time Pothole Detection	36
	8.2Hump Detection	36
	8.3GPS Integration	36
	8.4Driver Notification	36
	8.5Visual Alerts	36
	8.6Audio Alerts	36
	8.7Mobile App Integration	36
	8.8Crowd-Sourced Data	36
	8.9Severity Classification	36
	8.10Historical Data	37
9	**RESULTS AND DISCUSSIONS**	**38**
10	**CONCLUSION**	**40**
11	**REFERENCES**	**41**
12	**PSUEDOCODE**	**43**
13	**SCREENSHOTS**	**49**
14	**ENCLOSURES**	**51**

CHAPTER-1

INTRODUCTION

In the realm of modern transportation infrastructure, the maintenance of roads is a critical factor that directly influences both the safety of commuters and the overall economic well-being of a nation. Potholes and humps on roadways pose significant challenges, leading to potential accidents, vehicle damages, and hindrances to smooth traffic flow. Addressing this issue head-on, our project introduces an advanced system for the timely detection and notification of road anomalies, specifically focusing on potholes and humps.

Our innovative solution employs cutting-edge technology to provide a comprehensive approach to road maintenance. By utilizing ultrasonic sensors, we can accurately identify potholes and measure their depth, offering crucial data for efficient maintenance strategies. Simultaneously, our system incorporates RF transmitters strategically placed on road humps, communicating seamlessly with RF receivers within vehicles. As a vehicle approaches a hump, the system triggers instant notifications, alerting drivers through audible signals and visual warnings on an LCD display.

What sets our advanced potholes and humps notification system apart is its commitment to cost-effectiveness without compromising on efficiency. The integration of low-cost ultrasonic sensors ensures a practical and scalable solution that can be readily implemented in various settings. Furthermore, our system leverages the power of the Internet of Things (IoT) by transmitting real-time data to the cloud via WIFI, accessible to drivers through a dedicated mobile application.

In this era of interconnected smart technologies, our project aligns with the transformative potential of IoT, contributing to a safer and more efficient road network. Through proactive detection and immediate alerts, our system aims to enhance road safety, prevent accidents, and streamline the overall experience for drivers. Join us on this journey towards smarter, safer, and more sustainable road infrastructure – where advanced pothole and hump notifications pave the way for a smoother and safer tomorrow..

1.1 Definitions, Acronyms and Abbreviations :

1.1.1 IoT (Internet of Things) : A network comprising interconnected computing devices, mechanical and digital machines, objects, animals, or people. Each entity is equipped with unique identifiers and the capability to exchange data over a network without necessitating direct human-to-human or human-to-computer interaction.

1.1.2 Cloud Computing : A broad term denoting the provision of hosted services over the internet. Cloud computing allows companies to access and utilize computing resources, such as virtual machines (VMs), storage, or applications, as a utility – similar to electricity. This eliminates the need for organizations to construct and manage in-house computing infrastructures.

1.1.3 Arduino : An electronics platform operating on open-source principles and characterized by user-friendly hardware and software. Arduino boards have the capacity to interpret diverse inputs, including light on a sensor, a button press, or a Twitter message, and transform them into outputs, such as activating a motor or illuminating an LED. Users can program the microcontroller on the board by transmitting a set of instructions.

1.1.4 Blynk App : Blynk is a platform offering iOS and Android applications designed for controlling devices like Arduino and Raspberry Pi over the internet. It provides a digital dashboard where users can construct a graphical interface for their projects through a simple drag-and-drop mechanism for widgets. The setup is straightforward, enabling users to engage in experimentation in less than 5 minutes. Blynk is versatile, not bound to specific hardware, and supports various options. Whether connected through Wi-Fi, Ethernet, or innovative components like the ESP8266 chip, Blynk facilitates online connectivity for projects in the realm of the Internet of Things.

CHAPTER-2

LITERTURE SURVEY

He Youquan, Wang Jian, QiuHanxing, Zhang Wei, XieJianfang, “A Research of Pavement Potholes Detection Based on Three-Dimensional Project Transformation”, In Proceedings of International Congress on Image and Signal Processing, pp.1805-1808, 2011.**

In response to the increasing frequency of road accidents caused by inadequate road maintenance and reckless driving, this project introduces a system aimed at alerting drivers near sensitive areas such as schools, hospitals, and crowded zones. The system also offers real-time warnings by autonomously identifying potholes and humps on roads. The identification of crowded zones involves two units: a transmitter unit relaying zone-based information to drivers and a receiver unit equipped with an LCD for displaying this information. Ultrasonic sensors detect the presence of potholes and humps, signaling drivers through a buzzer. This comprehensive system not only aids in regulating vehicle speed in crowded areas but also helps prevent potential damage to vehicles.

MirceaStrutu, GrigoreStamatescu, Dan Popescu, “A Mobile Sensor Network Based Road Surface Monitoring System”, In Proceedings of IEEE Conference on System Theory, Control and Computing, pp.630–634, 2013.

This paper introduces a road surface defect identification system employing 3D accelerometers, GPS, and video modules deployed on vehicles. The architecture of the mobile platform and the central data aggregation algorithm are discussed, addressing the wireless communication coverage problem over large outdoor areas. The paper emphasizes the significance of the collected information by making it accessible to users through a GIS platform.

Moazzam, K. Kamal, S. Mathavan, S. Usman, M. Rahman, “Metrology and Visualization of Potholes using the Microsoft Kinect Sensor”, In Proceedings of IEEE Conference on Intelligent Transport System, pp.1284-1291, 2013.

In the realm of transportation engineering, detecting pavement distress and wear is crucial. This paper explores the use of a low-cost Kinect sensor to capture pavement depth images from concrete and asphalt roads. Meshes are generated for improved visualization of potholes, and the paper delves into the analysis of pothole area concerning depth. The methodology proposed in the paper aims to characterize potholes by estimating their area, length, width, and approximate volume through pavement image analysis, providing valuable insights for efficient maintenance strategies.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

While significant strides have been made in the development of advanced potholes and humps notification systems, there are still notable research gaps that present opportunities for further exploration and improvement. The following areas highlight key gaps in the existing research:

3.1 Accuracy Improvement :
Many current systems may lack precision in identifying potholes and humps, leading to false positives or negatives. Enhancing the accuracy of detection is a critical research area.

3.2 Real-time Data Processing :
Some methods might face challenges in processing data in real-time, impacting the timely notification of road hazards to drivers. Exploring efficient real-time processing techniques is essential.

3.3 Adaptability to Road Conditions:
Current systems may struggle to adapt to diverse road conditions, weather, and lighting variations. Research should focus on developing robust algorithms that perform well under different circumstances.

3.4 Cost-effective Sensor Integration:
Investigating cost-effective sensor solutions that can be easily integrated into existing road infrastructure is crucial for widespread implementation andscalability.

3.5 Integration with Autonomous Vehicles:
Future research should explore the seamless integration of pothole and hump detection systems with autonomous vehicles to enhance road safety and navigation.

3.6 User-Friendly Notifications:
Understanding the optimal ways to notify drivers about road hazards without causing distraction or confusion is vital for user acceptance and safety.

3.7 Network Security and Reliability:

Ensuring the security and reliability of the communication network between the road infrastructure, sensors, and drivers is essential to prevent malicious interference and ensure system dependability.

3.8 Long-Term System Maintenance:

Research could focus on developing systems that require minimal maintenance over extended periods to ensure sustainability and avoid frequent disruptions.

Addressing these research gaps would contribute to the development of more effective and reliable Automatic Detection and Notification systems for road hazards.

CHAPTER-4

PROPOSED METHODOLOGY

Block diagram:

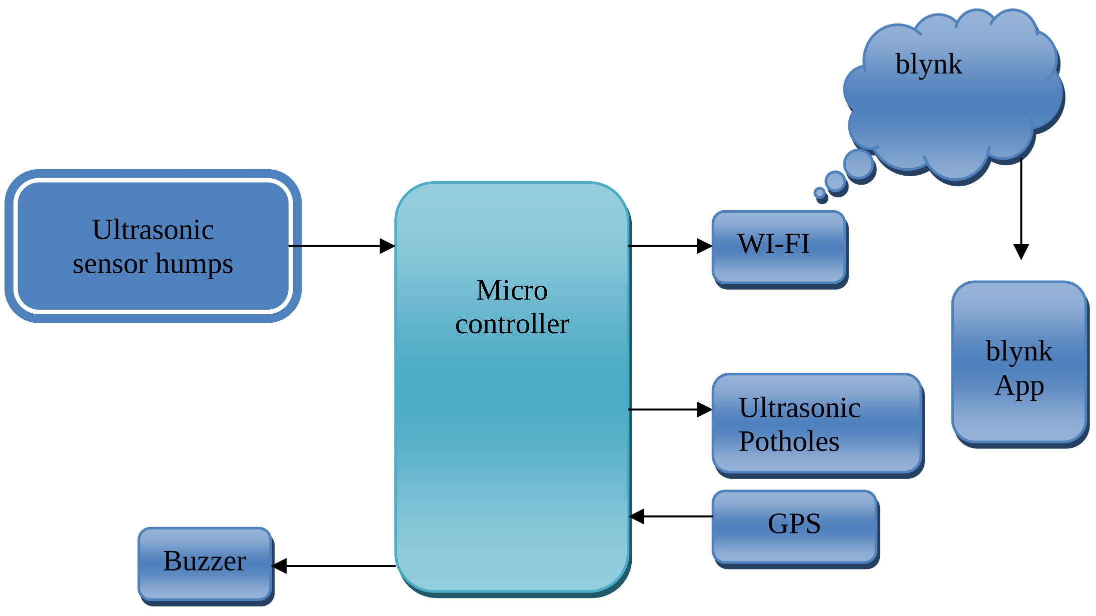


Fig 4.1

Components Used:

Hardware Requirements:

- Microcontroller
- Ultrasonic sensors
- Buzzer
- Ultrasonic sensor for potholes
- Ultrasonic sensor for humps
- WIFI

- Android Smart phone

Software Requirements:

- Programming language: Embedded C
- Software: Arduino IDE
- Blynk widget

CHAPTER-5

OBJECTIVES

- To analysis literature on the sensors used for detecting the potholes and humps.
- To analysis literature on microcontrollers used in the existing methods.
- To know about the Blynk app interfacing, how to use it for accessing information.
- To know about the Arduino board and Atmega328 microcontroller.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 ARDUINO :

At the core of this project lies the Arduino, a pivotal component orchestrating all operations. The interconnection of sensors, data transfer to the PC, and overall control are seamlessly managed by the Arduino.

Arduino, an open-source electronics prototyping platform, stands as a versatile and user-friendly foundation for this project. Tailored for artists, designers, hobbyists, and anyone keen on crafting interactive objects or environments, Arduino boasts flexible hardware and software. The boards excel at interpreting analog or digital input signals from an array of sensors, translating them into diverse outputs such as motor activation, LED control, cloud connectivity.

Programming Arduino is facilitated through the Arduino IDE, where users can send sets of instructions to the microcontroller on the board, a process commonly referred to as uploading software. Unlike its predecessors, Arduino eliminates the need for an additional piece of hardware (a programmer) to load new code onto the board – a simple USB cable suffices.

Adding to its accessibility, the Arduino IDE employs a simplified version of C++, lowering the barriers for individuals learning to program. Furthermore, Arduino offers a standardized form factor, encapsulating the microcontroller's functions within a user-friendly package.

In essence, Arduino acts as a microcontroller on the circuit board, streamlining the reception of inputs and the driving of outputs. Inputs, such as temperature sensors, motion sensors, and distance sensors, interact harmoniously with outputs like lights, screens, and motors. Serving as a small, programmable computer, Arduino empowers users to read and control various electrical components connected to it..

ARCHITECTURE OF CLOUD

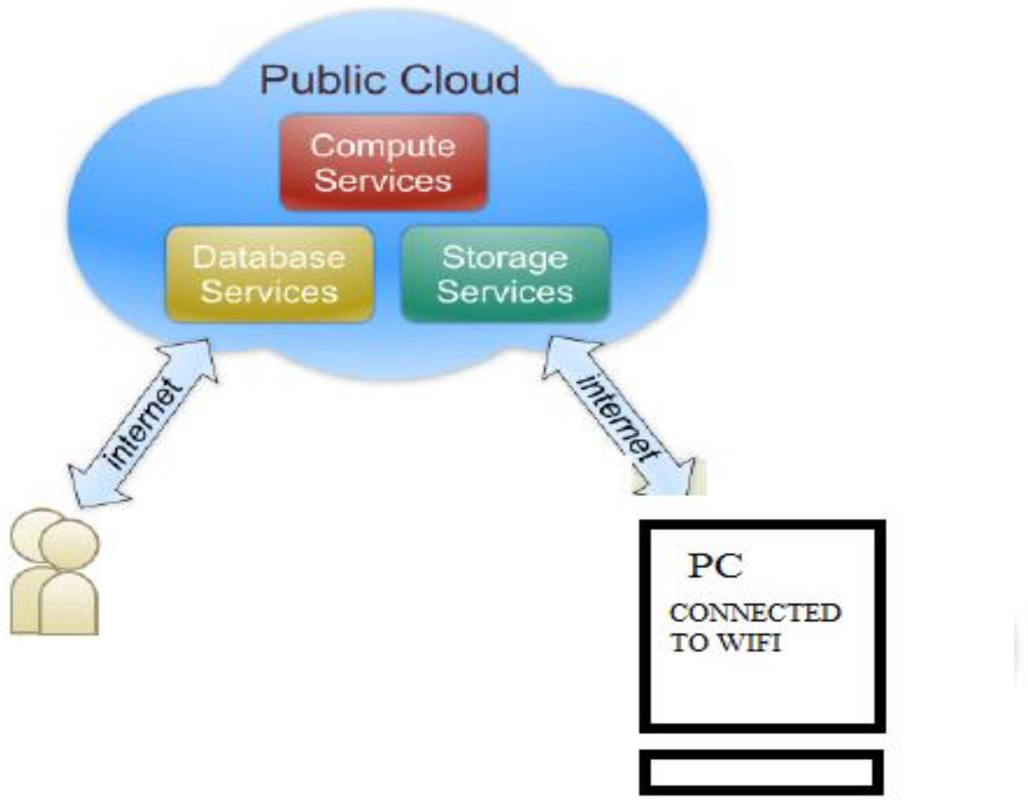


Fig 6.1

The Internet of Things (IoT), also sometimes referred to as the Internet of Everything (IoE), consists of all the web-enabled devices that collect, send and act on data they acquire from their surrounding environments using embedded sensors, processors and communication hardware.

These devices, often called "connected" or "smart" devices, can sometimes talk to other related devices, a process called **machine-to-machine** (M2M) communication, and act on the information they get from one another. Humans can interact with the gadgets to set them up, give them instructions or access the data, but the devices do most of the work on their own without human intervention. Their existence has been made possible by all the tiny mobile components that are available these days, as well as the always-online nature of our home and business networks.Connected devices also generate massive amounts of Internet traffic, including loads of data that can be used to make the devices useful, but can also be mined for other purposes. All this new data, and the Internet-accessible nature of the devices, raises both privacy and security concerns.

But this technology allows for a level of real-time information that we've never had before.

We can monitor our homes and families remotely to keep them safe. Businesses can improve processes to increase productivity and reduce material waste and unforeseen downtime. Sensors in city infrastructure can help reduce road congestion and warn us when infrastructure is in danger of crumbling. Gadgets out in the open can monitor for changing environmental conditions and warn us of impending disasters.

In a cloud computing system, there's a significant workload shift. Local computers no longer have to do all the heavy lifting when it comes to running applications. The network of computers that make up the cloud handles them instead. Hardware and software demands on the user's side decrease.

The only thing the user's computer needs to be able to run is the cloud computing system's**interface software**, which can be as simple as a Web browser, and the cloud's network takes care of the rest.

There's a good chance you've already used some form of cloud computing. If you have an e-mail account with a Web-based e-mail service like Hotmail, Yahoo! Mail or Gmail, then you've had some experience with cloud computing. Instead of running an e-mail program on your computer, you log in to a Web e-mail account remotely. The software and storage for your account doesn't exist on your computer -- it's on the service's computer cloud.

6.2 BLYNK APP

Blynk emerges as a versatile toolset, catering to makers, inventors, designers, educators, and tech enthusiasts seeking to leverage their smartphones for controlling electronics like Arduino, RaspberryPi, and similar devices. The Blynk platform simplifies the complexities of establishing internet connections, app development, and hardware coding.

Users can effortlessly assemble an impressive interface from a variety of widgets provided by Blynk. By uploading the example code to their hardware, enthusiasts can witness tangible results within a mere 5 minutes. This user-friendly approach caters to both beginners and seasoned makers, saving significant time for those with extensive technical expertise.

Blynk ensures compatibility with popular boards and shields, offering users the flexibility to integrate it seamlessly into existing or new projects. The platform's compatibility extends to both iOS and Android operating systems, ensuring a broad user base.

Blynk Cloud adds to the convenience, providing a free and open-source solution. Users can envision a prototyping board on their smartphones, where they can drag and drop buttons, sliders, displays, graphs, and other functional widgets. In just a few minutes, these widgets can control Arduino and retrieve data from it, making the development process efficient and user-friendly.

Notably, Blynk doesn't limit itself to a specific shield; rather, it's designed to support a wide range of boards and shields already in use. It operates over USB as well, allowing users to connect it to their laptops or desktops while awaiting internet shields to arrive.

Crucially, Blynk functions over the Internet, requiring only that the hardware can communicate with the Internet. Whether utilizing Ethernet, Wi-Fi, or the popular ESP8266, Blynk's libraries and example sketches facilitate online connectivity, allowing seamless pairing with smartphones and enabling a diverse range of projects.

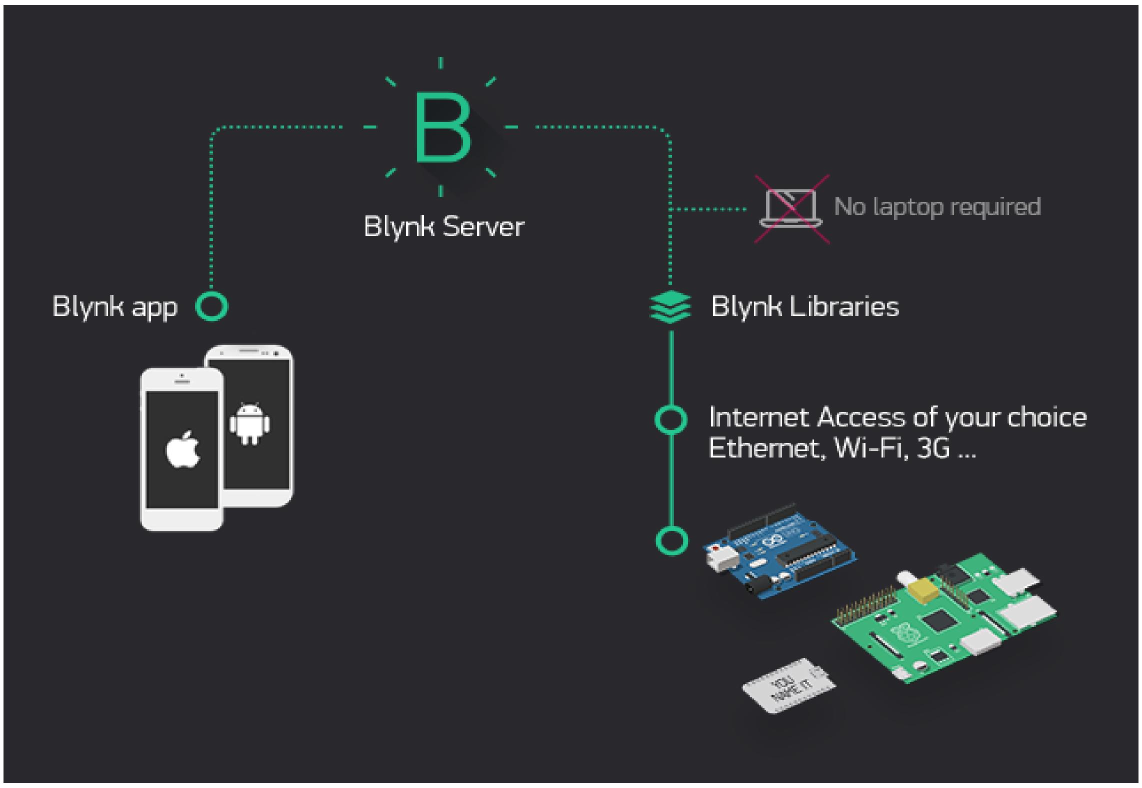


FIG: 6.2.1 Blynk architecture

Currently, Blynk libraries work with:

- USB
- Ethernet shield
- WiFi shield
- Arduino with Ethernet
- Arduino YÚN (testing in progress)
- ESP8266
- Raspberry Pi (Blynk will communicate with Pi's GPIOs)
- more Arduino compatible shields and boards (this list will be updated as we test the compatibility)

It's not that easy to take Arduino out of your home network, so we've built a Blynk server. It handles all the authentication and communication, and also keeps an eye on your board while the smartphone is offline. Blynk server runs on Java and is open-source. You will be able to run it locally if you really need to. Messaging between mobile apps , Blynk Server and Arduino is based on a simple, lightweight and fast binary protocol over TCP/IP sockets.

CREATING A PROJECT IN BLYNK APP

After downloading the app, create an account and log in. Welcome to Blynk!

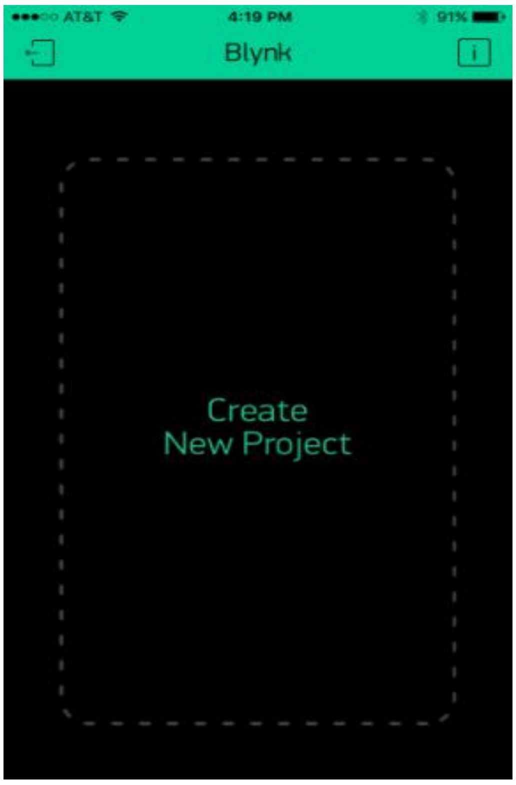
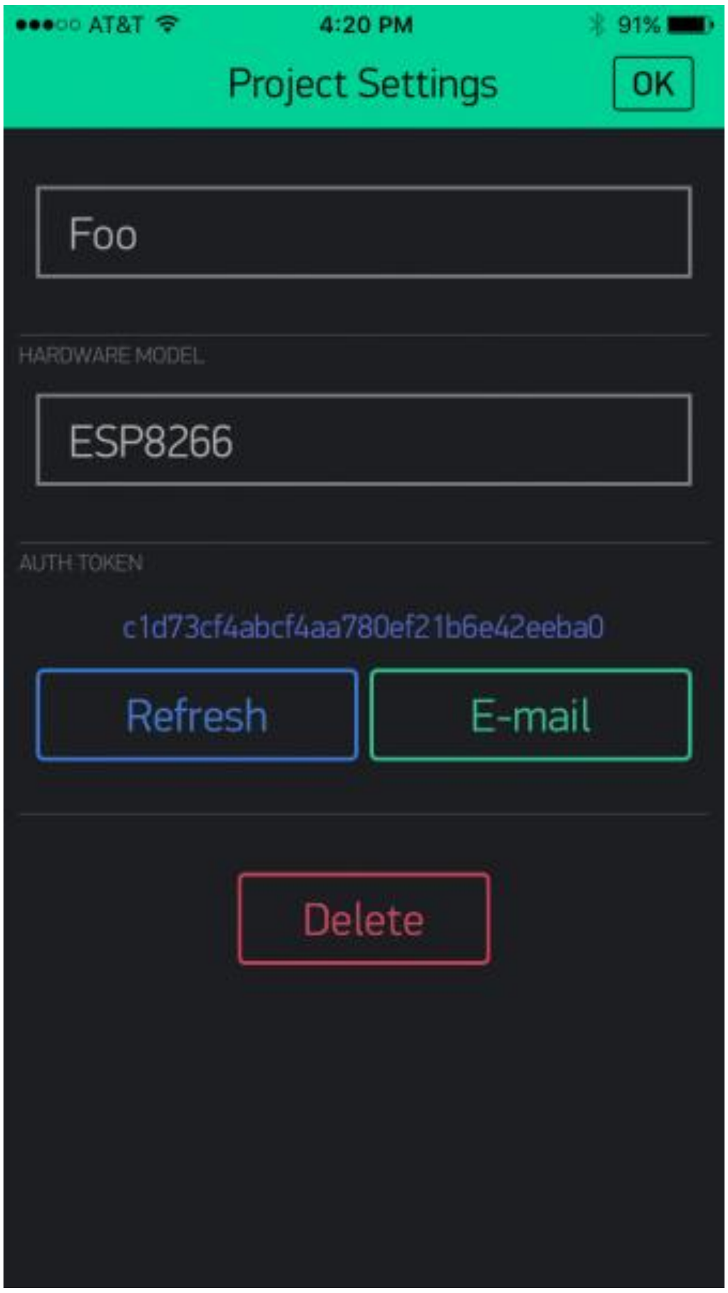


Fig 6.2.1

You'll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from Blynk'sGitHub repo, and follow along with the directions there to install the required libraries.

Create a Blynk Project

Next, click the "Create New Project" in the app to create a new Blynk app. Give it any name you please, just make sure the "Hardware Model" is set to **ESP8266.**



●●●○○ AT&T 4:20 PM 91%

Project Settings OK

Foo

HARDWARE MODEL

ESP8266

AUTH TOKEN

c1d73cf4abcf4aa780ef21b6e42eeba0

Fig :6.2.2

The **Auth Token** is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it to yourself.

Add Widgets to the Project

Then you'll be presented with a blank new project. To open the widget box, click in the project window to open.

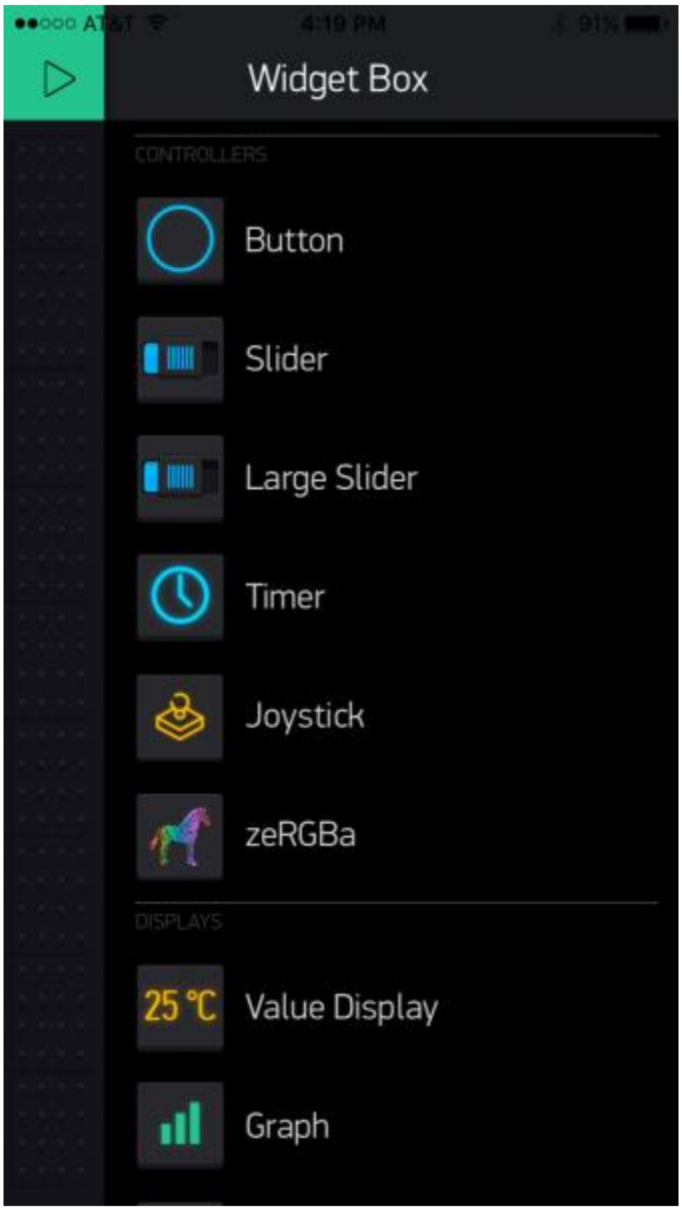
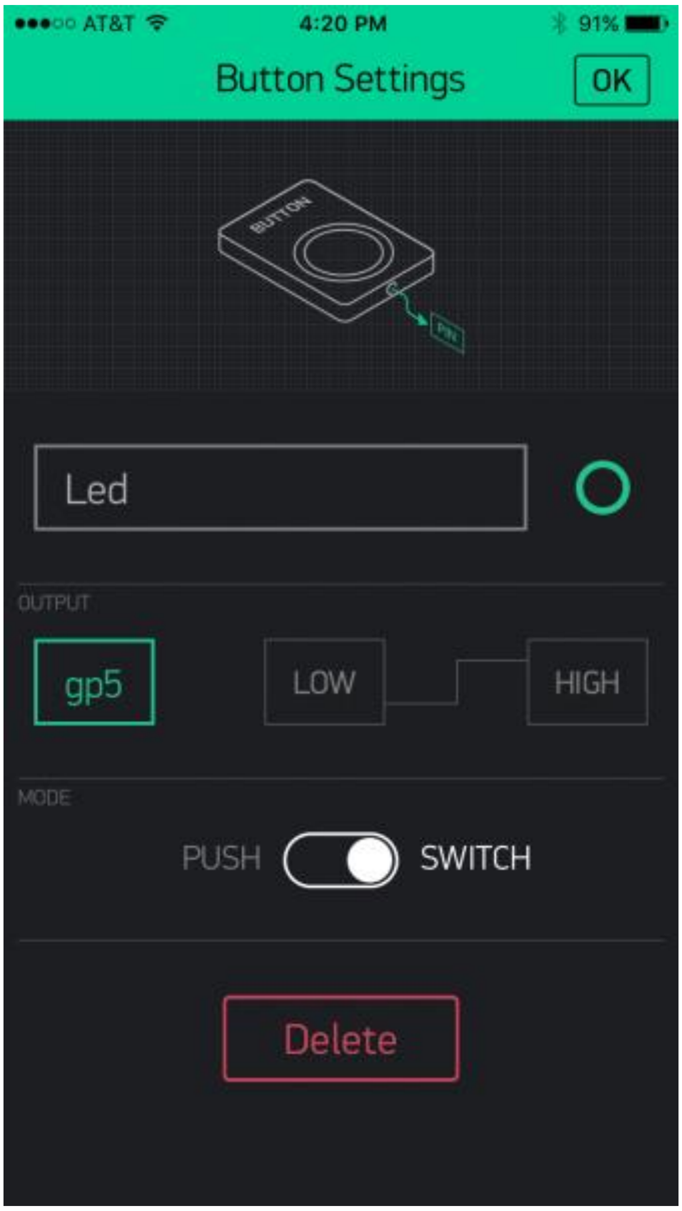


Fig :6.2.3

Add a **Button**, then click on it to change its settings. Buttons can toggle outputs on the ESP8266. Set the button's output to **gp5**, which is tied to an LED on the Thing Dev Board. You may also want to change the action to "Switch."


●●●○○ AT&T 4:20 PM 91%

Button Settings OK

Led

OUTPUT: gp5 LOW HIGH

MODE: PUSH SWITCH

Delete

Fig :6.2.4

Upload the Blynk Firmware

Now that your Blynk project is set up, open Arduino and navigate to the ESP8266_Standalone example in the File>Examples>Blynk>BoardsAndShields menu.

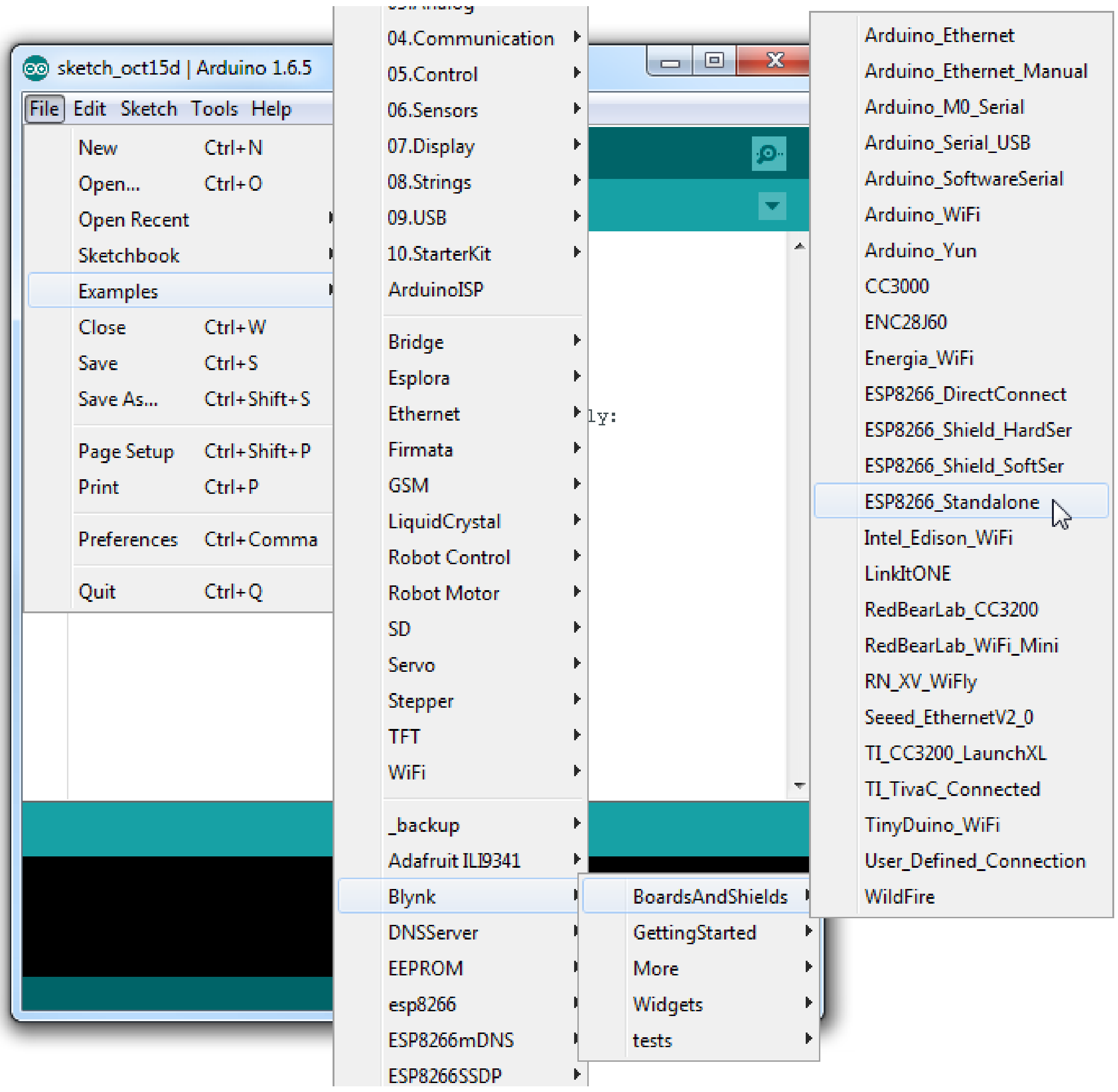


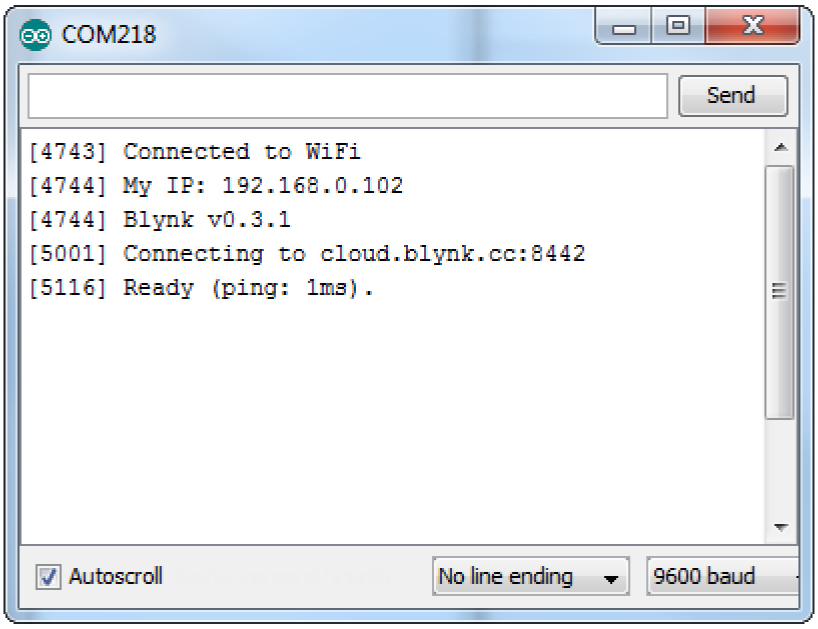
Fig : 6.2.5

Before uploading, make sure to paste your **authoriazation token** into the auth[] variable. Also make sure to **load your WiFi network settings into the Blynk.begin(auth, "ssid", "pass")** function.

Then upload!

Run the Project

After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the "Ready (ping: xms)." message.



```
[4743] Connected to WiFi
[4744] My IP: 192.168.0.102
[4744] Blynk v0.3.1
[5001] Connecting to cloud.blynk.cc:8442
[5116] Ready (ping: 1ms).
```

Fig :6.2.6

Then click the "Run" button in the top right corner of the Blynk app. Press the button and watch the LED!

Fig :6.2.7

Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.

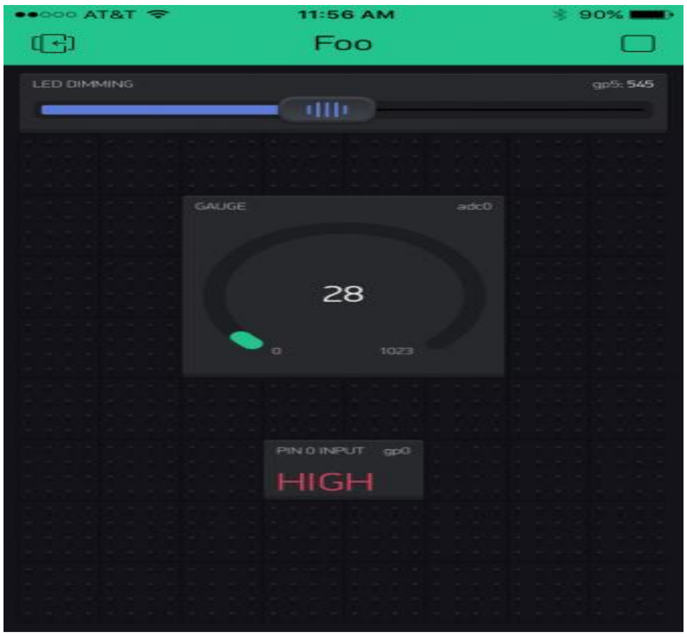


Fig :6:2.8

You can add analog output sliders, digital input monitors, and analog input gauges.

Overview

Arduino stands as an innovative open-source electronics platform, offering a user-friendly combination of hardware and software. Functioning as the brain of countless projects, Arduino boards excel at converting diverse inputs—ranging from light on a sensor to a Twitter message—into tangible outputs like motor activation, LED illumination, or online publishing. Users communicate their instructions to the microcontroller on the board using the Arduino programming language (based on Wiring) and the Arduino Software (IDE), which is built on Processing.

Over the years, Arduino has evolved into a versatile platform embraced by a global community of makers encompassing students, hobbyists, artists, programmers, and professionals. Their collective contributions have cultivated a vast repository of accessible knowledge, catering to both novices and experts. Originating as a tool for rapid prototyping at the Ivrea Interaction

Design Institute, Arduino targeted students lacking a background in electronics and programming. As it gained widespread adoption, Arduino adapted to emerging needs, expanding from simple 8-bit boards to cater to IoT applications, wearables, 3D printing, and embedded environments.

The hallmark of Arduino's ethos lies in its complete openness. All Arduino boards are released as open-source hardware, allowing users to independently build and customize them according to specific requirements. The software is also open-source, continually growing through global user contributions.

Beyond its role as a project, Arduino operates as a comprehensive computer hardware and software company. It nurtures a vibrant user community and manufactures microcontroller kits, empowering individuals to construct digital devices and interactive objects capable of sensing and controlling the physical world. Embracing open-source principles, Arduino's products are distributed under licenses like the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), enabling the manufacture and distribution of Arduino boards and software by anyone. These boards are available both commercially in preassembled form and as DIY kits, embodying the ethos of accessibility and empowerment that defines the Arduino ecosystem.

The project's board designs use a variety of microprocessors and controllers. These systems provide sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. The microcontrollers are mainly programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2005 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy,[2] aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.
"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current.

Why Arduino ?

Arduino's widespread adoption in diverse projects and applications can be attributed to its simple and accessible user experience. Whether utilized by beginners or advanced users, the Arduino software strikes a balance between ease of use and flexibility. Compatible with Mac, Windows, and Linux, it has become a go-to tool for various purposes.

educational settings, both teachers and students leverage Arduino to construct low-cost scientific instruments, demonstrate principles of chemistry and physics, and delve into programming and robotics. Designers and architects utilize it to prototype interactive designs, while musicians and artists experiment with new musical instruments and installations. At Maker Faires, Arduino is a cornerstone, serving as the building block for numerous projects exhibited by makers.

The versatility of Arduino extends to its role as a key tool for learning. Its step-by-step instructions and online community engagement make it accessible to anyone, from children and hobbyists to artists and programmers. Arduino encourages tinkering and exploration, fostering a collaborative atmosphere within the Arduino community.

While various microcontrollers and platforms exist for physical computing, Arduino stands out for several reasons:

Affordability : Arduino boards are relatively inexpensive, making them a cost-effective choice compared to other microcontroller platforms. Even the least expensive Arduino module can be assembled by hand, and pre-assembled modules are priced under $50.

Cross-Platform Compatibility : The Arduino Software (IDE) is versatile, running seamlessly on Windows, Macintosh OSX, and Linux. This cross-platform capability sets it apart, considering that most microcontroller systems are often limited to Windows.

User-Friendly Programming Environment : The Arduino Software (IDE) is designed to be straightforward, catering to beginners while offering flexibility for advanced users. Its interface is built on the Processing programming environment, providing familiarity for students learning to program in that environment.

• Open Source and Expandable Software :

The Arduino software stands as a testament to open-source principles, presenting a toolset available for extension by proficient programmers. Published as open-source, the software allows for expansion through C++ libraries, providing a pathway for individuals to delve into the technical intricacies by transitioning from Arduino to the underlying AVR C programming language. Moreover, users have the flexibility to directly incorporate AVR-C code into their Arduino programs, showcasing the extensibility of the language.

• Open Source and Extensible Hardware :

Arduino's commitment to openness extends to its hardware, with board plans distributed under a Creative Commons license. This approach empowers adept circuit designers to create personalized versions of the module, enhancing and extending its functionalities. Even users with limited experience can construct a breadboard version of the module, offering insights into its workings and providing a cost-effective alternative.

6.3 Hardware :

Arduino's hardware aspect is grounded in open-source principles. The hardware reference designs, shared under a Creative Commons Attribution Share-Alike 2.5 license, are accessible on the Arduino website. Some versions of the hardware even provide layout and production files, contributing to the collaborative ethos. Notably, the source code for the Arduino IDE is released under the GNU General Public License, version 2. However, an official Bill of Materials for Arduino boards hasn't been publicly disclosed by the Arduino team.

While the hardware and software designs are freely available under copyleft licenses, there is a request from developers to reserve the use of the name "Arduino" exclusively for the official product. This stipulation ensures that derived works seeking to use the Arduino name must obtain permission. The official policy underscores the project's openness to incorporating contributions from others into the official product. Notably, commercially released Arduino-compatible products have navigated this requirement by adopting -duino name variants, respecting the exclusivity of the official Arduino brand.

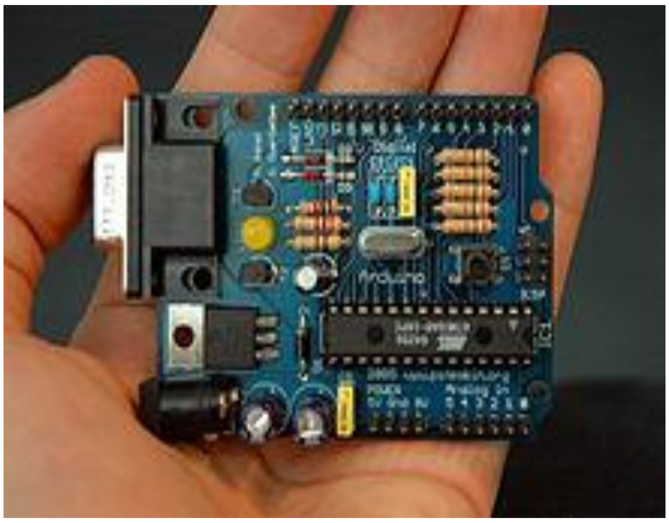


Fig :6.3.1

An early Arduino board[11] with an RS-232 serial interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

An Arduino board is essentially built around an Atmel microcontroller, available in 8-, 16-, or 32-bit AVR configurations. While the original Arduinos predominantly utilized Atmel megaAVR series chips, such as the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560, since 2015, there has been a shift to include microcontrollers from other manufacturers as well.

The Arduino's inherent versatility lies in its standardized connectors, allowing users to seamlessly attach the CPU board to a myriad of interchangeable add-on modules known as shields. Some shields establish communication directly with the Arduino through specific pins, while others are individually addressable via an I²C serial bus, enabling the stacking of multiple shields for parallel use.

Most Arduino boards come equipped with a 5 V linear regulator and a 16 MHz crystal oscillator (or, in some variants, a ceramic resonator). However, certain designs, like the LilyPad, operate at 8 MHz and forego the onboard voltage regulator due to form-factor restrictions.

A distinctive feature of Arduino is its pre-programmed microcontroller with a bootloader, streamlining the process of uploading programs to the on-chip flash memory. This sets Arduino apart from other devices that often require an external chip programmer. This design choice simplifies the user experience, as an ordinary computer can function as the programmer. As of now, the default bootloader installed on Arduino UNO boards is optiboot.

From a conceptual standpoint, the programming methodology for Arduino boards, when utilizing the Arduino integrated development environment, primarily involves a serial connection. The specifics of this implementation can vary based on the hardware version.

Certain serial Arduino boards incorporate a level shifter circuit, facilitating the conversion between RS-232 logic levels and transistor–transistor logic (TTL) level signals. In contemporary Arduino boards, programming is achieved through a Universal Serial Bus (USB) interface, utilizing USB-to-serial adapter chips like the FTDI FT232. Notably, some boards, including later models like the Uno, replace the FTDI chip with a distinct AVR chip containing USB-to-serial firmware. This AVR chip is reprogrammable via its own ICSP header.

In other variations, such as the Arduino Mini or the unofficial Boarduino, a detachable USB-to-serial adapter board or cable is employed. Additionally, alternatives like Bluetooth or other methods may be utilized. In scenarios where traditional microcontroller tools are applied instead of the Arduino IDE, standard AVR in-system programming (ISP) is employed.

This diverse range of programming approaches ensures compatibility with various hardware configurations, providing flexibility for users based on their specific needs and preferences.

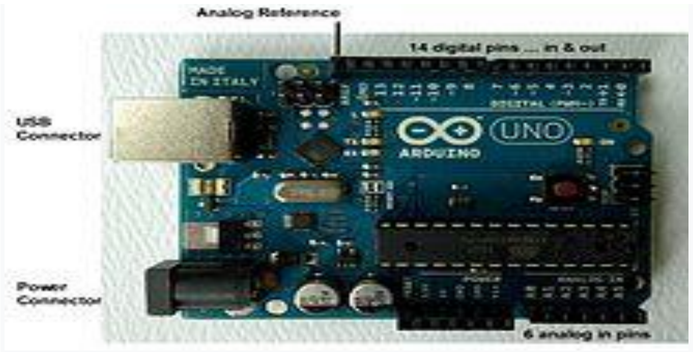


Fig :6.3.2

An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino board exposes a majority of the microcontroller's I/O pins, providing versatility for interfacing with other circuits. For instance, the Diecimila, Duemilanove, and the current Uno models offer 14 digital I/O pins, six of which support pulse-width modulated (PWM) signals. Additionally, there are six analog inputs that can also function as digital I/O pins. These pins are conveniently located on the top of the board, accessible via female 0.1-inch (2.54 mm) headers. Various plug-in application shields are commercially available, expanding the capabilities of the Arduino ecosystem. In some models like the Arduino Nano or the Boarduino, male header pins on the underside enable direct connection to solderless breadboards.

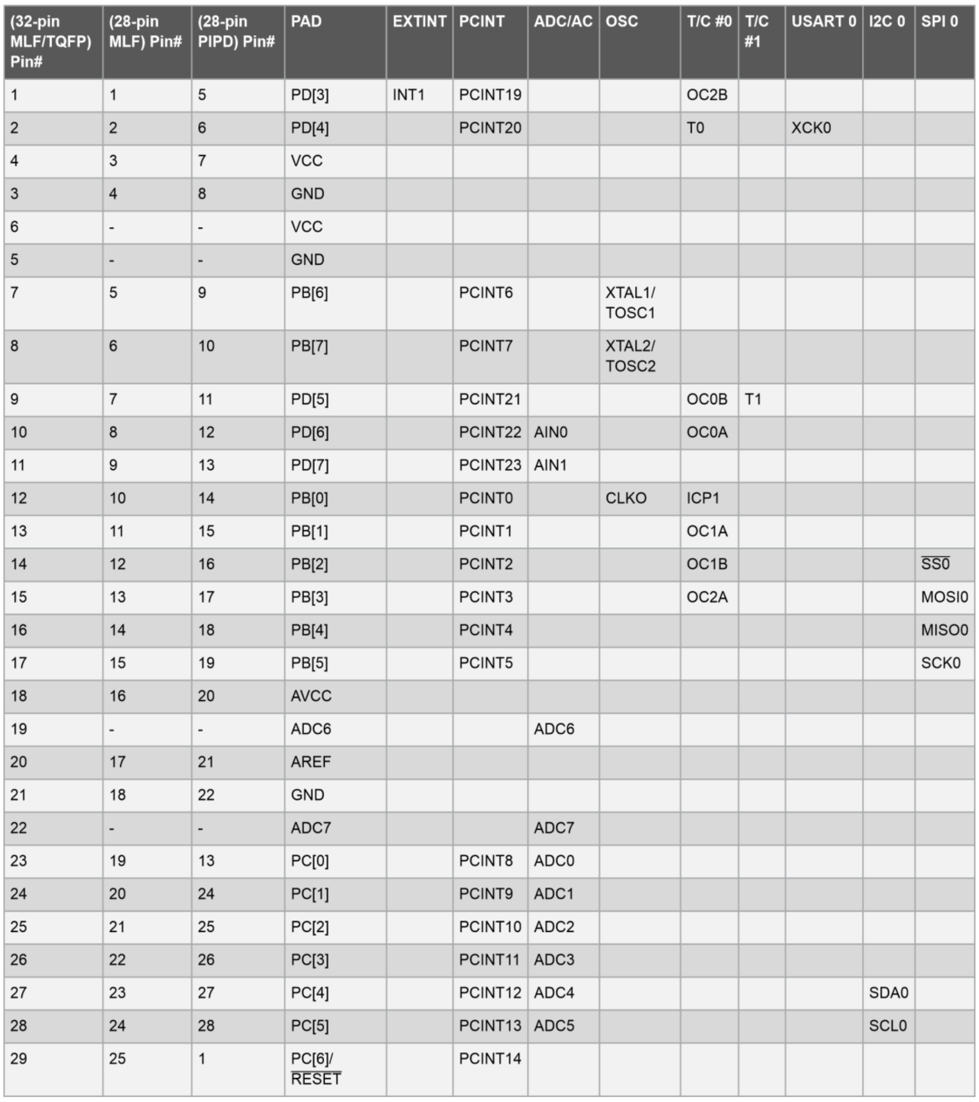
The Arduino landscape includes a plethora of boards, both Arduino-compatible and Arduino-derived, each offering unique features. Some boards maintain functional equivalence with the Arduino, allowing for interchangeable use. Others enhance the basic

Arduino by incorporating additional output drivers, particularly useful in educational settings for projects involving small robots and buggies..

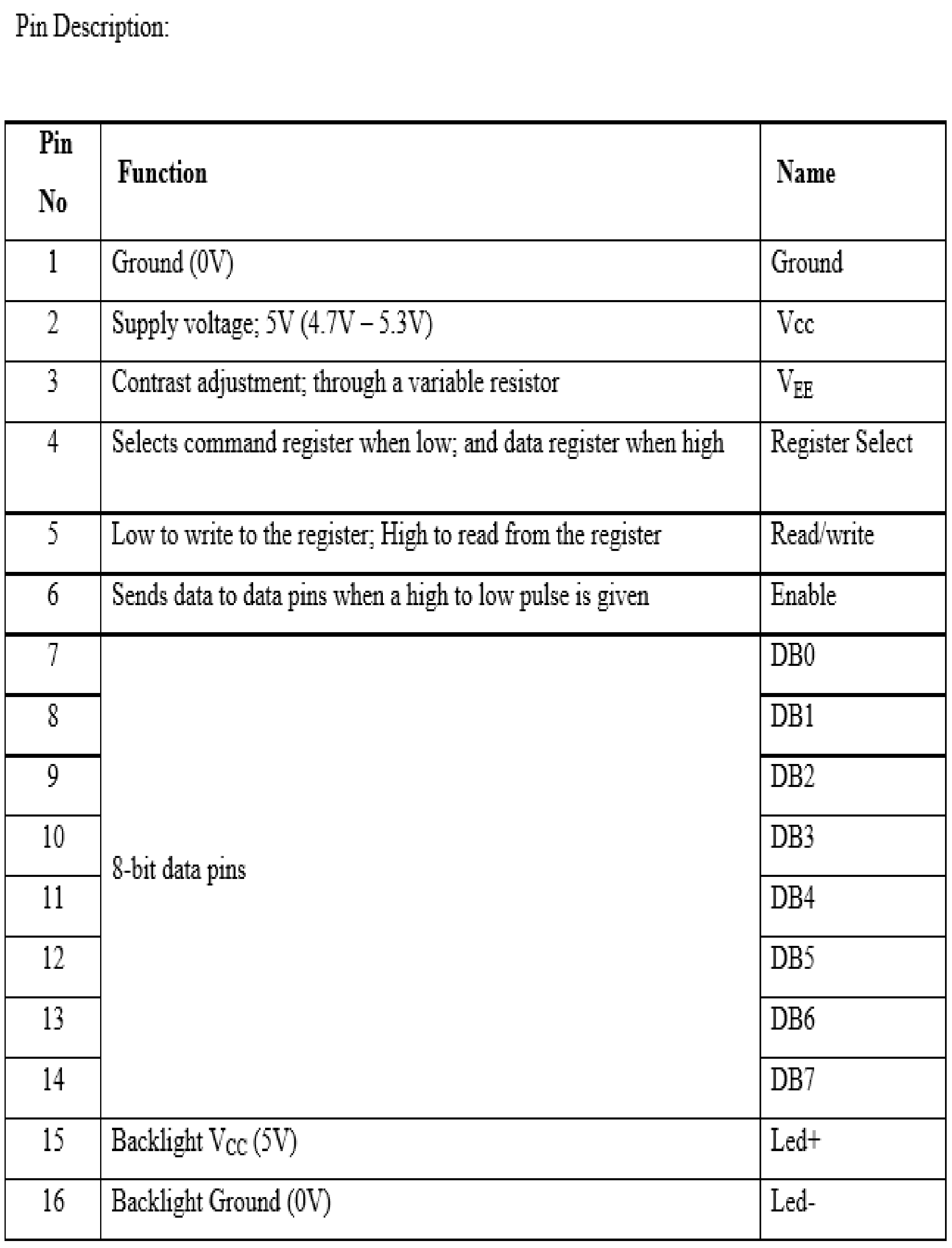
Regarding software development, the Arduino/Genuino Uno is programmable using the Arduino Software (IDE). The Tools > Board menu allows users to select "Arduino/Genuino Uno" based on the microcontroller on their board. The ATmega328 on the Arduino/Genuino

Uno is preprogrammed with a bootloader, enabling the uploading of new code without an external hardware programmer. The bootloader communicates using the original STK500 protocol.

PORT Function Multiplexing :



(32-pin MLF/TQFP) Pin#	(28-pin MLF) Pin#	(28-pin PIPD) Pin#	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C #0	T/C #1	USART 0	I2C 0	SPI 0
1	1	5	PD[3]	INT1	PCINT19			OC2B				
2	2	6	PD[4]		PCINT20			T0		XCK0		
4	3	7	VCC									
3	4	8	GND									
6	-	-	VCC									
5	-	-	GND									
7	5	9	PB[6]		PCINT6		XTAL1/ TOSC1					
8	6	10	PB[7]		PCINT7		XTAL2/ TOSC2					
9	7	11	PD[5]		PCINT21			OC0B	T1			
10	8	12	PD[6]		PCINT22	AIN0		OC0A				
11	9	13	PD[7]		PCINT23	AIN1						
12	10	14	PB[0]		PCINT0		CLKO	ICP1				
13	11	15	PB[1]		PCINT1			OC1A				
14	12	16	PB[2]		PCINT2			OC1B				$\overline{\text{SS0}}$
15	13	17	PB[3]		PCINT3			OC2A				MOSI0
16	14	18	PB[4]		PCINT4							MISO0
17	15	19	PB[5]		PCINT5							SCK0
18	16	20	AVCC									
19	-	-	ADC6			ADC6						
20	17	21	AREF									
21	18	22	GND									
22	-	-	ADC7			ADC7						
23	19	13	PC[0]		PCINT8	ADC0						
24	20	24	PC[1]		PCINT9	ADC1						
25	21	25	PC[2]		PCINT10	ADC2						
26	22	26	PC[3]		PCINT11	ADC3						
27	23	27	PC[4]		PCINT12	ADC4					SDA0	
28	24	28	PC[5]		PCINT13	ADC5					SCL0	
29	25	1	PC[6]/ $\overline{\text{RESET}}$		PCINT14							



Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

6.4 Digital Pins

The Arduino pins can be set up as either inputs or outputs. This explanation focuses on the behavior of pins in input mode. While the title mentions digital pins, it's important to note that the majority of Arduino's analog pins can be configured and used similarly to digital pins.

By default, Arduino pins are set as inputs, and there's no need to explicitly declare them as inputs using pinMode(). In this state, the pins are in a high-impedance state, meaning they have very little impact on the circuit they are sampling. It's akin to having a series resistor of 100 megohms in front of the pin. Due to this property, input pins demand extremely small amounts of current to transition from one state to another. This characteristic makes them useful for various tasks such as implementing capacitive touch sensors, reading an LED as a photodiode, or working with analog sensors using methods like RCTime.

However, it's crucial to note that pins configured as pinMode(pin, INPUT) without any connections or with unconnected wires may exhibit seemingly random state changes. They can pick up electrical noise from the environment or capacitively couple with the state of nearby pins.

To address this, pullup resistors can be employed with pins configured as INPUT. Pullup resistors help steer an input pin to a known state when no external input is present. These resistors are connected to +5V. Alternatively, pulldown resistors (resistors to ground) can be used for a similar purpose. A common value for pullup or pulldown resistors is 10K ohms.

The value of this pullup depends on the microcontroller used. On most AVR-based boards, the value is guaranteed to be between 20kΩ and 50kΩ. On the Arduino Due, it is between 50kΩ and 150kΩ. For the exact value, consult the datasheet of the microcontroller on your board.

When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed.

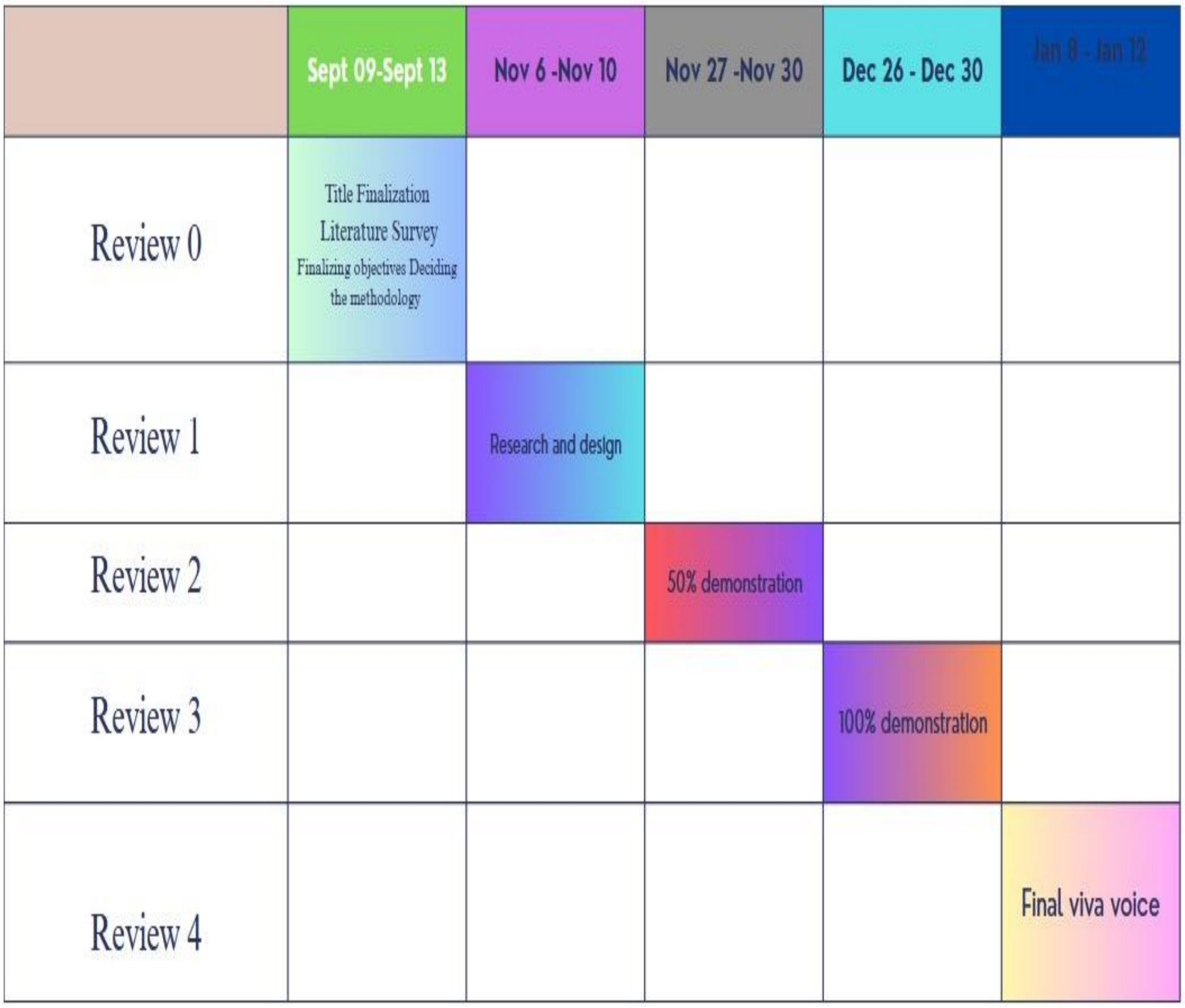
The pullup resistors provide enough current to dimly light an LED connected to a pin that has been configured as an input. If LEDs in a project seem to be working, but very dimly, this is likely what is going on.

The pullup resistors are controlled by the same registers (internal chip memory locations) that control whether a pin is HIGH or LOW. Consequently, a pin that is configured to have pullup resistors turned on when the pin is an INPUT, will have the pin configured as HIGH if the pin is then switched to an OUTPUT with pinMode(). This works in the other direction as well, and an output pin that is left in a HIGH state will have the pullup resistors set if switched to an input with pinMode()..

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)



	Sept 09-Sept 13	Nov 6 -Nov 10	Nov 27 -Nov 30	Dec 26 - Dec 30	Jan 8 - Jan 12
Review 0	Title Finalization Literature Survey Finalizing objectives Deciding the methodology				
Review 1		Research and design			
Review 2			50% demonstration		
Review 3				100% demonstration	
Review 4					Final viva voice

Fig :7.1

CHAPTER-8

OUTCOMES

8.1 Real-time Pothole Detection:

Implement a system that can automatically detect potholes on roads in real-time.

8.2 Hump Detection:

Extend the system to identify road humps for comprehensive road safety.

8.3 GPS Integration:

Integrate GPS data to provide accurate location information for detected potholes and humps.

8.4 Driver Notification:

Alert drivers about upcoming potholes and humps in advance to promote safe driving.

8.5 Visual Alerts:

Use visual indicators on the driver's interface to highlight the location and severity of detected road issues.

8.6 Audio Alerts:

Incorporate audible warnings to ensure drivers are alerted even if they're not actively looking at the interface.

8.7 Mobile App Integration:

Develop a mobile app to provide notifications to drivers on their smartphones.

8.8 Crowd-Sourced Data:

Allow drivers to report road issues, contributing to a dynamic and up-to-date database.

8.9 Severity Classification:

Categorize potholes and humps based on severity to prioritize alerts and notifications.

8.10 Historical Data:

Store historical data to analyze patterns and improve the accuracy of future predictions.

8.11 Adaptive System:

Implement machine learning algorithms to adapt and improve over time based on user feedback and evolving road conditions.

8.12 Emergency Services Integration:

Enable the system to alert emergency services in case of severe road conditions.

8.13 Offline Mode:

Ensure basic functionality even in areas with poor or no internet connectivity.

8.14 Multi-language Support:

Provide alerts and notifications in multiple languages to cater to diverse user groups.

8.15 Low-Resource Devices:

Optimize the system to run efficiently on devices with limited processing power.

CHAPTER-9
RESULTS AND DISCUSSIONS

- **Introduction :**
  Automatic detection and notification systems for potholes and humps on roads have been developed to enhance driver safety.

- **Sensor Technology :**
  Utilizes advanced sensor technologies, such as accelerometers and cameras, to detect irregularities in road surfaces.

- **Real-time Monitoring :**
  Constantly monitors road conditions in real-time to provide timely information to drivers.

- **Pothole Detection Algorithm :**
  Employs a sophisticated algorithm that analyzes sensor data to identify potholes accurately.

- **Hump Detection Algorithm :**
  Incorporates a separate algorithm for recognizing speed bumps or humps, ensuring comprehensive road surface analysis.

- **Notification System :**
  Notifies drivers through in-car displays or mobile applications when a pothole or hump is detected ahead.

- **Severity Assessment :**
  Provides information on the severity of detected road irregularities to help drivers anticipate the level of caution required**.**

- **Integration with Navigation Systems :**
  Seamlessly integrates with GPS navigation systems to provide precise location-based alerts.

- **Data Logging and Analytics :**
  Collects and logs data on road conditions, contributing to analytics that can aid in urban planning and road maintenance**.**

- **User Feedback Mechanism :**
  Establishes a feedback mechanism for users to report inaccuracies or provide additional information, improving system accuracy over time.

- **Cost-Effective Infrastructure Improvement :**
  Facilitates proactive road maintenance by identifying areas with frequent issues, potentially reducing long-term repair costs.

- **Reduced Accident Rates :**
  Aids in minimizing accidents caused by unexpected road conditions, contributing to overall road safety.

- **Environmental Impact :**
  Contributes to environmental conservation by reducing vehicle wear and tear and lowering the frequency of road repairs, subsequently decreasing resource consumption**.**

- **Social Impact :**
  Enhances the overall driving experience, promoting a safer and more comfortable transportation environment for all road users.

- **Challenges and Future Improvements :**
  Acknowledges potential challenges, such as false positives, and suggests ongoing improvements in sensor technology and algorithms to address these issues.

CHAPTER-10

CONCLUSION

10.1 In conclusion, the automatic detection and notification system for potholes and humps on roads represents a significant advancement in road safety and driver assistance. Through the integration of cutting-edge technologies such as computer vision and sensor networks, this system provides real-time information to drivers, enhancing their situational awareness and ultimately contributing to a safer driving experience.

10.2 By promptly identifying and notifying drivers about the presence of potholes and humps, the system minimizes the risk of accidents, vehicle damage, and discomfort for road users. This proactive approach not only improves overall road safety but also reduces the economic burden associated with repairs and maintenance of vehicles caused by road irregularities.

10.3 The implementation of this system aligns with the broader goals of smart cities and intelligent transportation systems. It fosters efficiency in traffic flow and promotes a seamless commuting experience by empowering drivers with crucial information about road conditions. Furthermore, the data collected by the system can be utilized for strategic urban planning, aiding authorities in identifying areas requiring infrastructural improvements.

10.4 The environmental impact is also noteworthy, as the reduction in vehicular accidents and traffic congestion leads to decreased fuel consumption and emissions. This aligns with global efforts to create sustainable and eco-friendly transportation solutions.

In summary, the automatic detection and notification of potholes and humps on roads represent a pivotal step towards creating safer, smarter, and more sustainable urban environments. As technology continues to evolve, further refinements and enhancements to such systems are expected, solidifying their role in revolutionizing the way we navigate and perceive our road networks.

REFERENCES

1. *India Transport Sector.* [Online]. Available: http://web.worldbank.org/WBSITE/EXTERNAL/COUNTRIES/SOUTHASIAEXT/EXTSARREGTOPTRANSPORT/0,,contentMDK:20703625~menuPK:868822~pagePK:34004173~piPK:34003707~theSitePK:579598,00.html

2. Rajeshwari S., SanthoshHebbar, Varaprasad G., "Implementing Intelligent Traffic Control System for Congestion Control, Ambulance Clearance and Stolen Vehicle Detection", IEEE Sensors Journal, Vol.15, No.2, pp.1109-1113, 2015

3. I. Moazzam, K. Kamal, S. Mathavan, S. Usman, M. Rahman, "Metrology and Visualization of Potholes using the Microsoft Kinect Sensor", *In Proceedings of IEEE Conference on Intelligent Transport System,* pp.1284-1291, 2013.

4. Sudarshan S. Rode, Shonil Vijay, PrakharGoyal, PurushottamKulkarni, KaviArya, "Pothole Detection and Warning System", *In Proceedings of International Conference on Electronic Computer Technology*, pp.286-290, 2009.

5. He Youquan, Wang Jian, QiuHanxing, Zhang Wei, XieJianfang, "A Research of Pavement Potholes Detection Based on Three-Dimensional Project Transformation", *In Proceedings of International Congress on Image and Signal Processing*, pp.1805-1808, 2011.

6. Jin Lin, Yayu Liu, "Potholes Detection Based on SVM in the Pavement Distress Image", *In Proceedings of International Symposium on Distributed Computing and Applications to Business, Engineering and Science,* pp.544-547,2010

7. Faith Orhan, P. ErhanEren, "Road Hazard Detection and Sharing with Multimodal Sensor Analysis on Smartphones", *In Proceedings of International Conference on Next Generation Mobile Apps, Services and Technologies,* pp. 56-61, 2013.

8. ArtisMednis, Girts Strazdins, ReinholdsZviedris, GeorgijsKanonirs, Leo Selavo, "Real Time Pothole Detection using Android Smartphones with Accelerometers", *In Proceedings of Distributed Computing in Sensor Systems Workshop,* pp.1-6, 2011.

9. Zhen Zhang, Xiao Ai, C. K. Chan and NaimDahnoun, *"*An Efficient Algorithm for Pothole Detection using Stereo Vision*", In Proceedings of IEEE Conference on Acoustic, Speech and Signal Processing,* pp.564-568, 2014.

10. MirceaStrutu, GrigoreStamatescu, Dan Popescu, “A Mobile Sensor Network Based Road Surface Monitoring System”, *In Proceedings of IEEE Conference on System Theory, Control and Computing,* pp.630–634, 2013.

11. SachinBharadwaj, Sundra Murthy, GollaVaraprasad “Detection of potholes in autonomous vehicle”, *IET Intelligent Transport Systems*, Vol.8, No.6, pp.543-549, 2013.

12. SandeepVenkatesh, AbhiramE,Rajarajeswari S, Sunil Kumar K M and ShreyasBalakuntala, “An Intelligent System to Detect, Avoid and Maintain Potholes: A Graph Theoretic Approach”, *In Proceedings of International Conference on Mobile Computing and Ubiquitous Networking,* pp.80, *2014.*

13. ShambhuHegde, Harish V. Mekali, GollaVaraprasad, “Pothole Detection and Inter vehicular Communication” *Technical Report of Wireless Communications Laboratory, BMS College of Engineering, Bangalore 19.*

14. The GPS website, *www.gpsinformation.org.*

15. Prachi More, SudhishSurendran, SayaliMahajan and Saurabh Kumar Dubey, “Potholes and pitfalls spotter”, IMPACT:IJRET, Vol 4, pp. 69-74, 2014.

16. X. Yu and E. Salari, “Pavement Pothole Detection severity Measurement using laser Imaging”, *In Proceedings of IEEE International conference on EIT,* pp.1-5, 2014.

17. Kongyang Chen, Mingming Lu, Xiaopeng Fan, Mingming Wei, and Jinwu Wu, “Road Condition Monitoring Using On-board Three-axis Accelerometer and GPS Sensor”, *In Proceedings of International ICST conference on Communication and Networking in China,* pp.1032-1037, 2011.

18. Fagen Li, Pan Xiong, “Practical Secure Communication for Integrating Wireless Sensor Networks Into the Internet of Things”, IEEE Sensor Journal, Vol.13, Issue 10, pp 3677-3684, 2013.

19. AlessioCarullo and Marco Parvis, "An Ultrasonic Sensor for Distance Measurement in Automotive Applications", IEEE Sensor Journal, Vol. 1, pp.143-147, 2001.

APPENDIX-A

PSUEDOCODE

```
#define BLYNK_TEMPLATE_ID "TMPLNMeFNlES"
#define BLYNK_TEMPLATE_NAME "potholes"
#define BLYNK_AUTH_TOKEN "-lc-2TMHmeUZIXP-j2VJd4DJNV63DHse"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include <TinyGPS++.h>
TinyGPSPlusgps;
static const uint32_t GPSBaud = 9600;

float latitude;
float longitude;

#define echoPin1 2
#define trigPin1 15
#define echoPin2 23
#define trigPin2 22

long duration;
float distance;

#define IN_1 5
#define IN_2 18
#define IN_3 19
#define IN_4 21


char auth[]="o1UHSb8NIhNKG6z9GZyO9K8ZlC4MZbnk";
char ssid[] = "Draj";
char pass[] = "no iwont tell";

int obstacle(int trigPinx, int echoPinx) {
digitalWrite(trigPinx, LOW);
delayMicroseconds(2);

digitalWrite(trigPinx, HIGH);
delayMicroseconds(10);
digitalWrite(trigPinx, LOW);

  duration = pulseIn(echoPinx, HIGH);

  distance = duration * 0.034 / 2;
  return distance;
}



void goRight(){
digitalWrite(IN_1, HIGH);
digitalWrite(IN_2, LOW);
digitalWrite(IN_3, HIGH);
digitalWrite(IN_4, LOW);
  }

void goLeft(){
digitalWrite(IN_1, LOW);
digitalWrite(IN_2, HIGH);
digitalWrite(IN_3, LOW);
digitalWrite(IN_4, HIGH);
  }

void goForword(){
```

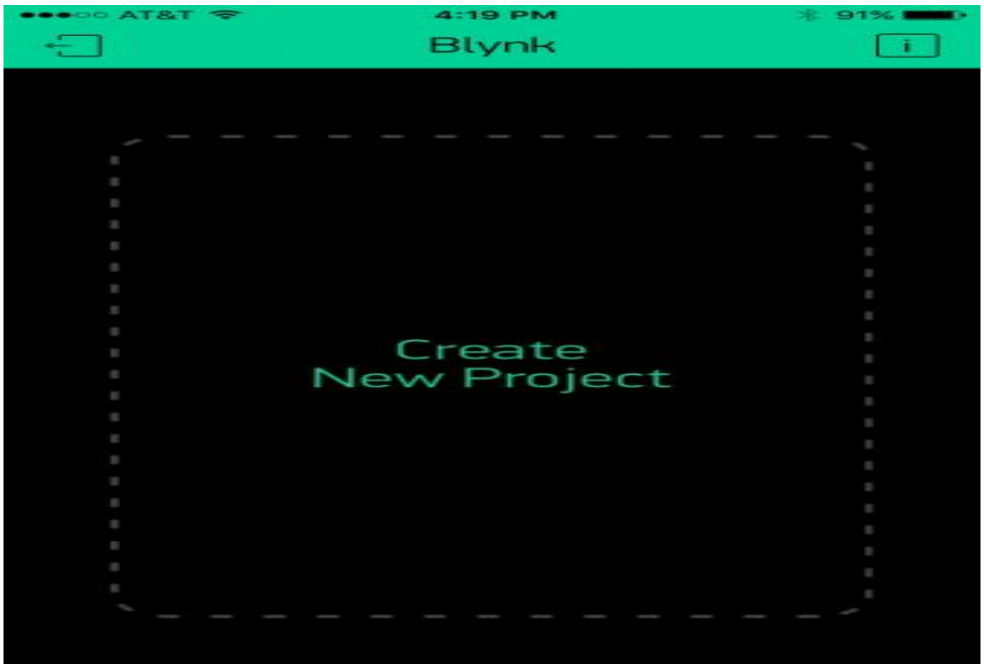
```
digitalWrite(IN_1, HIGH);
digitalWrite(IN_2, LOW);
digitalWrite(IN_3, LOW);
digitalWrite(IN_4, HIGH);
  }

void goBack(){
digitalWrite(IN_1, LOW);
digitalWrite(IN_2, HIGH);
digitalWrite(IN_3, HIGH);
digitalWrite(IN_4, LOW);
  }

void hold(){
digitalWrite(IN_1, LOW);
digitalWrite(IN_2, LOW);
digitalWrite(IN_3, LOW);
digitalWrite(IN_4, LOW);
 }

BLYNK_WRITE(V6)
{
  int pinValue1 = param.asInt();
  if (pinValue1 == 1){goForword();
Serial.print("F");}
  else hold();
}

BLYNK_WRITE(V7)
{
  int pinValue2 = param.asInt();
  if (pinValue2 == 1){goBack();
Serial.print("B");}
  else hold();
```

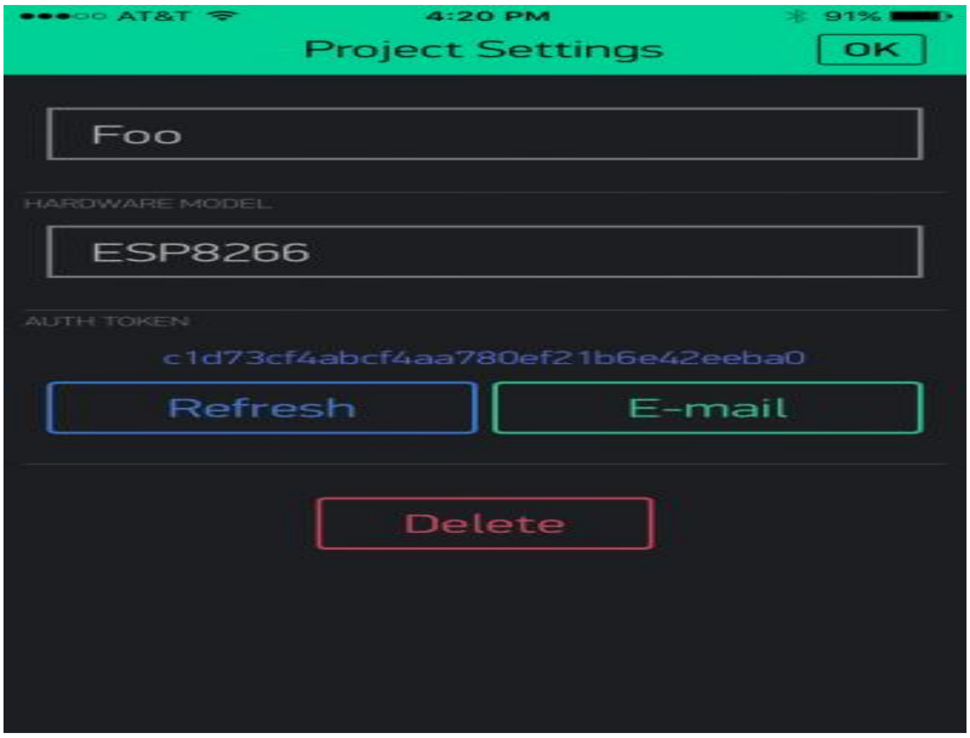
```
}

BLYNK_WRITE(V8)
{
  int pinValue3 = param.asInt();
  if (pinValue3 == 1){goLeft();
Serial.print("L");}
  else hold();
}

BLYNK_WRITE(V9)
{
  int pinValue4 = param.asInt();
  if (pinValue4 == 1){goRight();



Serial.print("R");}
  else hold();
}

void setup()
{
Serial.begin(9600);   // Initiate a serial communication
Serial.begin(GPSBaud);
Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(IN_1, OUTPUT);
pinMode(IN_2, OUTPUT);
pinMode(IN_3, OUTPUT);
pinMode(IN_4, OUTPUT);
}
```

```
void GPS()
{
if (gps.charsProcessed() < 10)
  {
    //Serial.println("No GPS detected: check wiring.");

  // Blynk.virtualWrite(V4, "GPS ERROR");         // Value Display widget  on V4 if GPS not
detected
  }

}
void displaygpsInfo()
{

  if (gps.location.isValid() )
  {

     latitude = (gps.location.lat());      //Storing the Lat. and Lon.
     longitude = (gps.location.lng());

Serial.print("LAT:  ");
Serial.println(latitude, 6);                // float to x decimal places

Serial.print("LONG: ");
Serial.println(longitude, 6);

      }
}

void loop()
{
            while (Serial.available() > 0)
    {
     // sketch displays information every time a new sentence is correctly encoded.
     if (gps.encode(Serial.read()))
displaygpsInfo();
  }

Blynk.run();
  int dist1 = obstacle(trigPin1,echoPin1);
  int dist2 = obstacle(trigPin2,echoPin2);

Serial.print(dist1);
Serial.print("\t");
Serial.println(dist2);
String  long_lat = String(float(latitude))+", "+String(float(longitude));
  String hump = "Hump Detected " + long_lat;
  String pothole = "Pothole Detected " + long_lat;
  if (dist1 <3){
Blynk.virtualWrite(V10,hump);
  }
  if (dist2 >6){
Blynk.virtualWrite(V10,pothole);
  }
else{
Blynk.virtualWrite(V10,"          ");
 }
}
```

APPENDIX-B

SCREENSHOTS





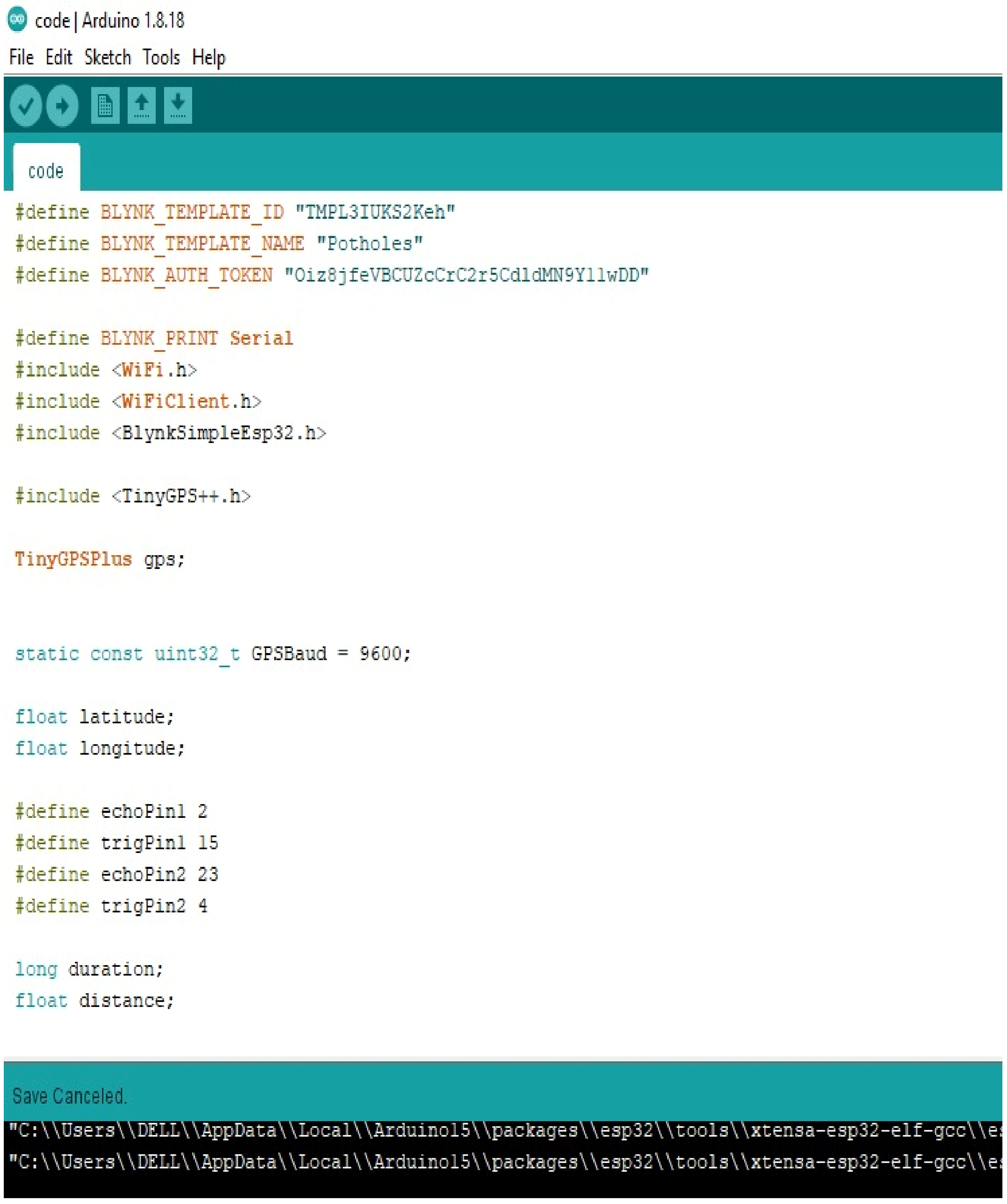
Foo

HARDWARE MODEL

ESP8266

AUTH TOKEN

c1d73cf4abcf4aa780ef21b6e42eeba0



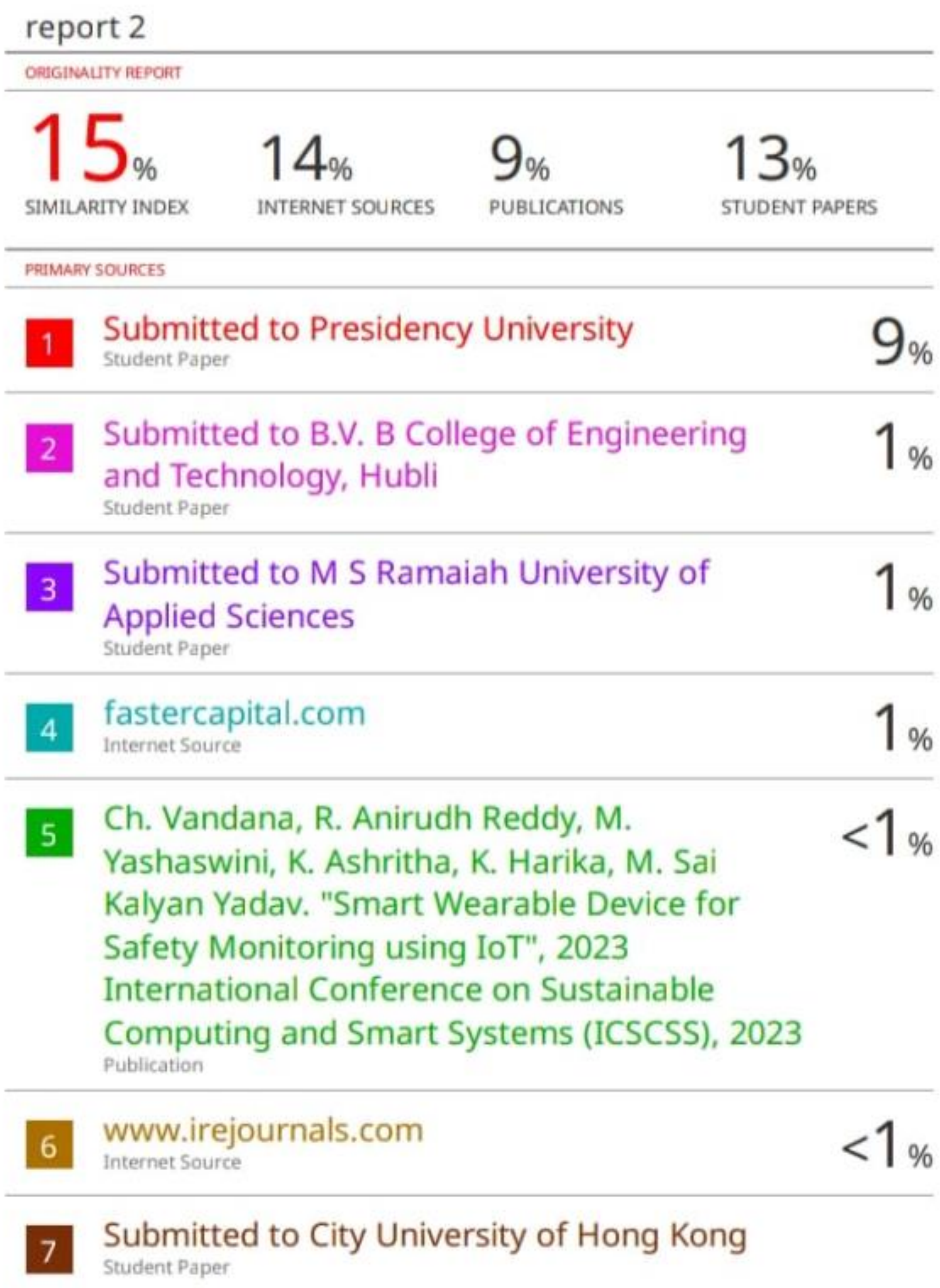
```
#define BLYNK_TEMPLATE_ID "TMPL3IUKS2Keh"
#define BLYNK_TEMPLATE_NAME "Potholes"
#define BLYNK_AUTH_TOKEN "Oiz8jfeVBCUZcCrC2r5CdldMN9YllwDD"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include <TinyGPS++.h>

TinyGPSPlus gps;


static const uint32_t GPSBaud = 9600;

float latitude;
float longitude;

#define echoPin1 2
#define trigPin1 15
#define echoPin2 23
#define trigPin2 4

long duration;
float distance;
```

```
Save Canceled.
"C:\\Users\\DELL\\AppData\\Local\\Arduino15\\packages\\esp32\\tools\\xtensa-esp32-elf-gcc\\e
"C:\\Users\\DELL\\AppData\\Local\\Arduino15\\packages\\esp32\\tools\\xtensa-esp32-elf-gcc\\e
```

APPENDIX-C

ENCLOSURES

Similarity Index / Plagiarism Check report clearly showing the Percentage(15%). No need of page-wise explanation.



report 2

ORIGINALITY REPORT

15%	14%	9%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	9%
2	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	1%
3	Submitted to M S Ramaiah University of Applied Sciences Student Paper	1%
4	fastercapital.com Internet Source	1%
5	Ch. Vandana, R. Anirudh Reddy, M. Yashaswini, K. Ashritha, K. Harika, M. Sai Kalyan Yadav. "Smart Wearable Device for Safety Monitoring using IoT", 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), 2023 Publication	<1%
6	www.irejournals.com Internet Source	<1%
7	Submitted to City University of Hong Kong Student Paper	

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

The Project work carried out here is mapped to SDG-9

The project we carried out here contributes to industry, innovation and infrastructure, It can find Potholes and Hump and Road to Drive safety.