Institut Supérieur d'Informatique
et de Multimédia de Gabes

Internship report

Internet of Things (IoT) and Embedded Systems

Prepared by : **RAKIA SOUEI**

Higher Institute of Computer Science and Multimedia of Gabes

Realised In: 2024/ 2025

Table of Contents

I. Introduction to Emertxe

Emertxe is a premier training institute based in India, Since 2003, specializing in embedded systems and full-stack development. Established with the vision of bridging the gap between academic learning and industry requirements, Emertxe provides hands-on training programs tailored to equip students and professionals with practical skills in technology and software development.

Training Programs and Specializations

Emertxe offers a range of specialized courses, including:

- **Embedded Systems Training**: Focuses on microcontrollers, real-time operating systems, Linux kernel programming, and IoT applications.
- **Full-Stack Development**: Covers front-end and back-end web technologies, including JavaScript, Node.js, React, and databases.
- **Internship and Placement Programs**: Provides industry-oriented training, enabling students to work on real-world projects and gain practical exposure.

Industry Collaboration and Placement Assistance

Emertxe has established strong industry collaborations with leading companies in the tech sector. Their training methodology emphasizes hands-on learning, live projects, and mentorship by industry experts.

II. Introduction to the Internet of Things (IoT)

Introduction to IoT

The **Internet of Things (IoT)** represents a transformative shift in how we interact with technology and the world around us. IoT refers to the network of physical objects—devices, vehicles, appliances, and other items—embedded with sensors, software, and connectivity that enable them to collect and exchange data. This interconnected ecosystem allows for the seamless flow of information between devices, systems, and humans, creating new opportunities for innovation, efficiency, and insight.

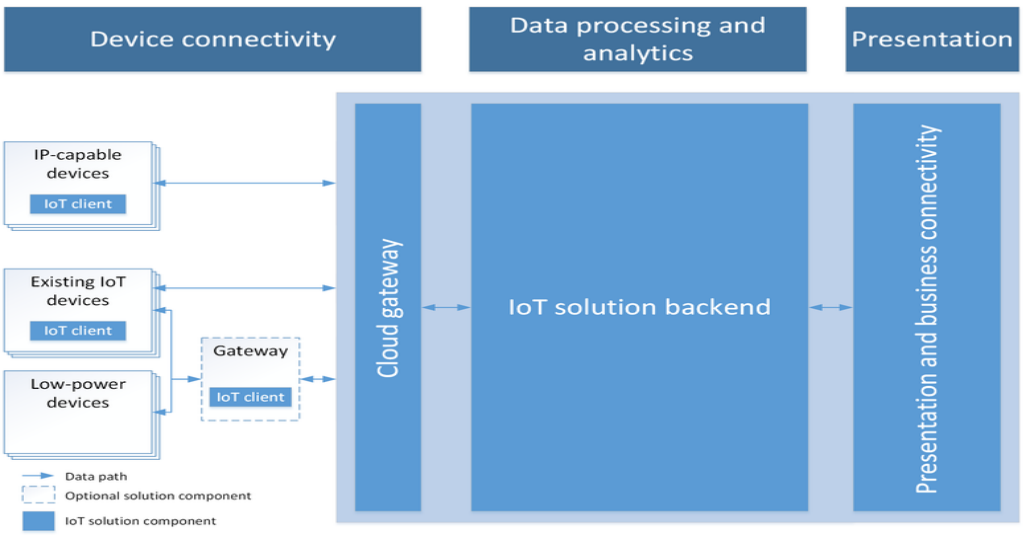
IoT is not just about connecting devices to the internet; it is about leveraging the data generated by these devices to drive actionable insights and improve decision-making. As of today, there are **over 13.8 billion IoT devices globally**, outnumbering the world's population by nearly twice. This rapid growth is driven by the increasing adoption of IoT across various industries, including **utilities, agriculture, manufacturing, transportation, and consumer electronics**. The global IoT market is projected to reach **$21.3 billion in 2022**, reflecting a **22% increase** from the previous year, according to Gartner.

The promise of IoT lies in its ability to merge perspectives between devices, systems, and humans, enabling a deeper understanding of the world around us. By tying together insights with action, IoT has the potential to revolutionize industries, improve quality of life, and address some of the world's most pressing challenges.

IoT Architecture and Technical Requirements

At the heart of IoT lies its **architecture**, which serves as the foundation for building scalable, secure, and efficient IoT systems. IoT architecture is complex and multifaceted, requiring a combination of hardware, software, connectivity, and data analytics to function effectively. The architecture can be broken down into several key components:

1. **Devices and Sensors**:
   These are the physical objects embedded with technology to collect data from their environment. Examples include smart thermostats, wearable devices, and industrial sensors.
2. **Connectivity**:
   IoT devices rely on various communication protocols (e.g., Wi-Fi, Bluetooth, Zigbee, LoRaWAN) to transmit data to other devices or central systems. The choice of connectivity depends on factors such as range, power consumption, and data transfer speed.
3. **Data Processing and Analytics**:
   Once data is collected, it needs to be processed and analyzed to extract meaningful insights. This can occur at the **edge** (on the device itself) or in the **cloud**, depending on the application.
4. **User Interface**:
   The insights generated by IoT systems are often presented to users through dashboards, mobile apps, or other interfaces, enabling them to make informed decisions.
5. **Security and Privacy**:
   Given the sensitive nature of the data collected by IoT devices, security is a critical component of IoT architecture. Measures such as encryption, authentication, and access control are essential to protect against cyber threats.



III. Introduction to Embedded Systems

Introduction to Embedded Systems

Embedded systems are specialized computing systems that are designed to perform specific tasks within larger mechanical or electrical systems. Unlike general-purpose computers, which are capable of running a wide variety of applications, embedded systems are tailored to execute dedicated functions, often with real-time computing constraints. These systems are ubiquitous in modern technology, found in everything from household appliances and consumer electronics to industrial machines and automotive systems.

The definition of an embedded system, as highlighted in the presentation, is:
"Any combination of hardware and software which is intended to do a specific task can be called as an embedded system."
This definition underscores the fundamental nature of embedded systems: they are purpose-built to perform a single or a set of specific functions efficiently and reliably.

Embedded systems are categorized into several types, including **stand-alone systems**, **real-time systems**, **networked systems**, and **mobile systems**. Each category serves different applications and environments, but all share the common characteristic of being designed for a specific purpose. For example, a **stand-alone embedded system** might be a microwave oven, while a **networked embedded system** could be a smart thermostat connected to the internet.

The importance of embedded systems cannot be overstated. They are the backbone of modern technology, enabling the functionality of countless devices and systems that we rely on daily. From medical devices that monitor patients' vital signs to automotive systems that control engine performance, embedded systems play a critical role in ensuring the reliability, efficiency, and safety of these technologies.







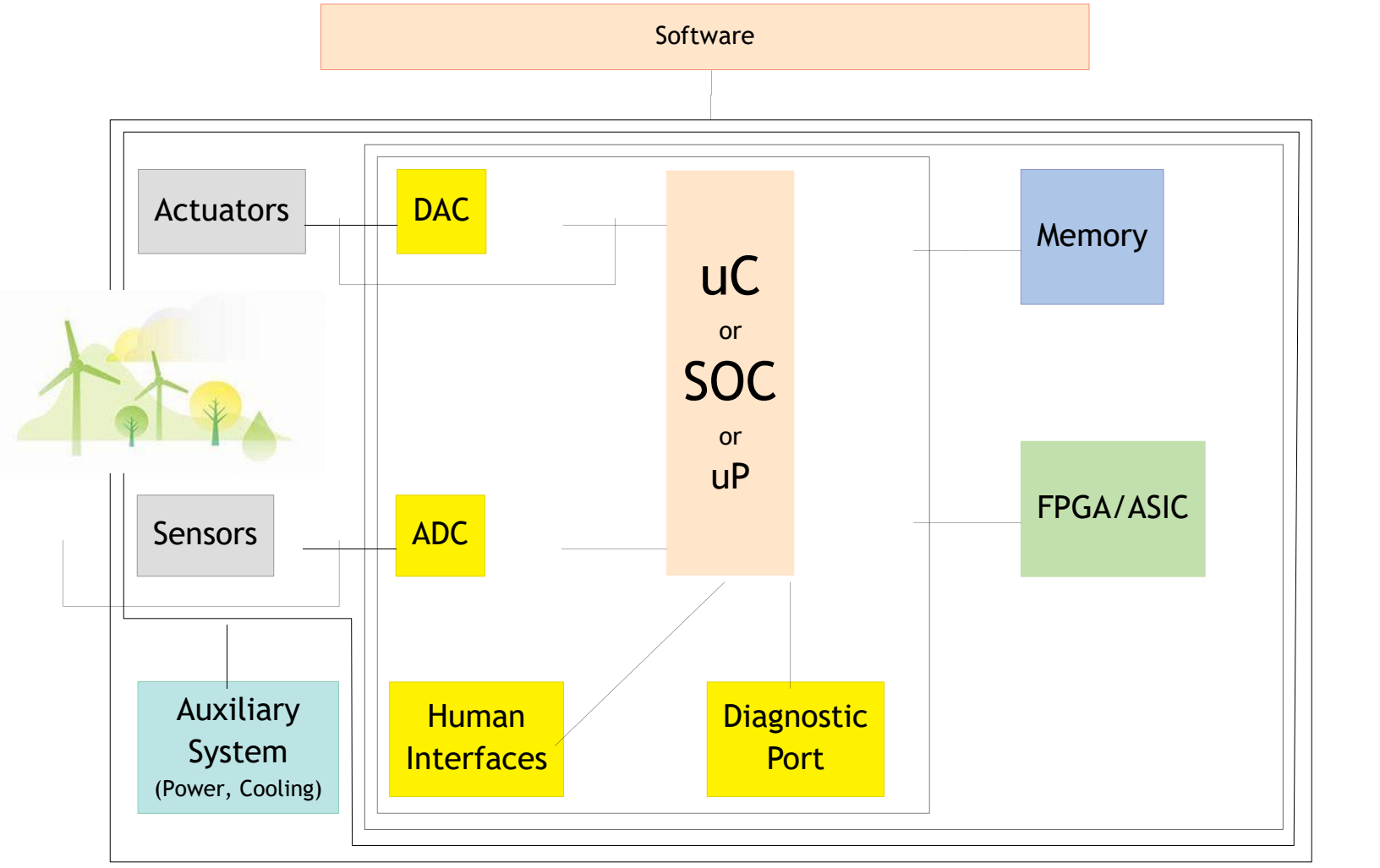

Components of Embedded Systems

Embedded systems are composed of several key components that work together to achieve their intended functionality. These components can be broadly categorized into **hardware** and **software** elements, each playing a crucial role in the system's operation.

1. **Hardware Components**:
    - **Microcontroller (uC) or System on Chip (SoC)**:
      The microcontroller or SoC is the brain of the embedded system. It processes input data, executes software instructions, and controls the system's operations. Microcontrollers are often used in simpler systems, while SoCs are employed in more complex applications due to their integrated processing power and peripherals.
    - **Memory**:
      Embedded systems require memory to store both the software (firmware) and the data they process. This includes **ROM** for storing the firmware and **RAM** for temporary data storage during operation.
    - **Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC)**:
      These components are essential for systems that interact with the physical world. ADCs convert analog signals (e.g., from sensors) into digital data that the microcontroller can process, while DACs perform the reverse operation, converting digital signals into analog outputs (e.g., for actuators).
    - **FPGA/ASIC**:
      Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) are used in systems that require high-speed processing or specialized functionality. FPGAs are reconfigurable, making them versatile for prototyping, while ASICs are custom-designed for specific tasks.
    - **Sensors and Actuators**:
      Sensors collect data from the environment (e.g., temperature, pressure, motion), while actuators perform actions based on the system's output (e.g., turning on a motor, opening a valve).
    - **Auxiliary Systems**:
      These include power supplies, cooling systems, and other supporting components that ensure the embedded system operates reliably.
2. **Software Components**:

- **Firmware**:
  The software that runs on the embedded system's microcontroller or SoC. It is typically written in low-level programming languages like C or assembly to optimize performance and resource usage.
- **Human Interfaces**:
  Many embedded systems include user interfaces, such as buttons, touchscreens, or displays, to allow users to interact with the system.
- **Diagnostic Ports**:
  These are used for debugging, testing, and updating the system's software.

The integration of these hardware and software components is what enables embedded systems to perform their specific tasks efficiently and reliably. The design of an embedded system must carefully balance performance, power consumption, cost, and size, depending on the application.



Challenges and Future Trends in Embedded Systems

While embedded systems are highly effective in their specific roles, they also face several challenges, particularly as technology continues to evolve. One of the primary challenges is **complexity**. As embedded systems become more advanced, they often require more sophisticated hardware and software, increasing the complexity of design, development, and testing. This complexity is further compounded by the need for **real-time performance** in many applications, where delays in processing can lead to system failures or safety hazards.

Another significant challenge is **power consumption**. Many embedded systems, especially those in mobile or battery-powered devices, must operate efficiently to conserve energy. This requires careful optimization of both hardware and software to minimize power usage without compromising performance.

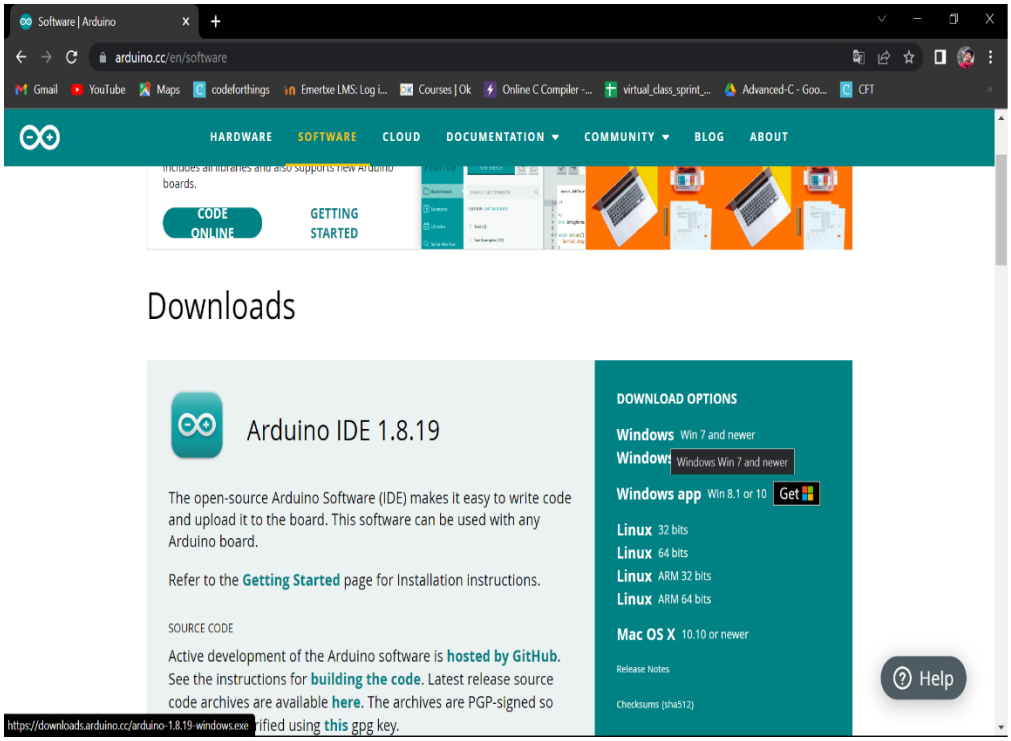
Security is also a growing concern in embedded systems, particularly as more devices become connected to the internet (e.g., IoT devices). Embedded systems are often vulnerable to cyberattacks due to their limited processing power and lack of robust security features. Ensuring the security of embedded systems requires the implementation of encryption, secure boot processes, and regular software updates.

Looking to the future, embedded systems are expected to become even more integrated with emerging technologies such as **artificial intelligence (AI)**, **machine learning (ML)**, and **5G connectivity**. AI and ML can enhance the capabilities of embedded systems by enabling them to make intelligent decisions based on data, while 5G connectivity will allow for faster and more reliable communication between devices. Additionally, the rise of **edge computing** will see more processing being done locally on embedded systems, reducing latency and bandwidth usage.

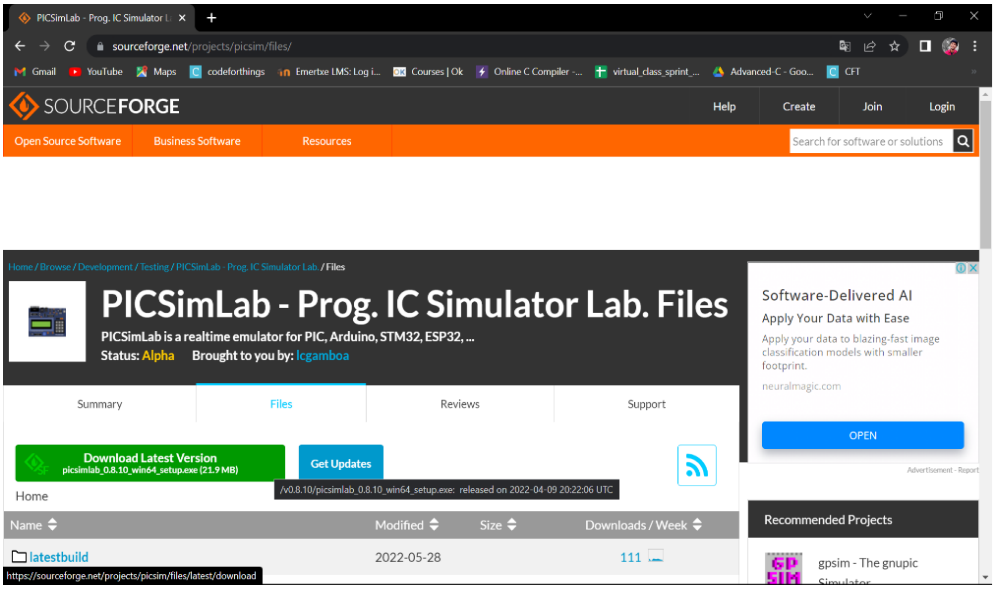
In conclusion, embedded systems are a cornerstone of modern technology, enabling the functionality of countless devices and systems across various industries. As technology continues to advance, embedded systems will play an increasingly important role in driving innovation and improving efficiency. However, addressing challenges such as complexity, power consumption, and security will be essential to unlocking the full potential of these systems.

IV Softwares used for the project

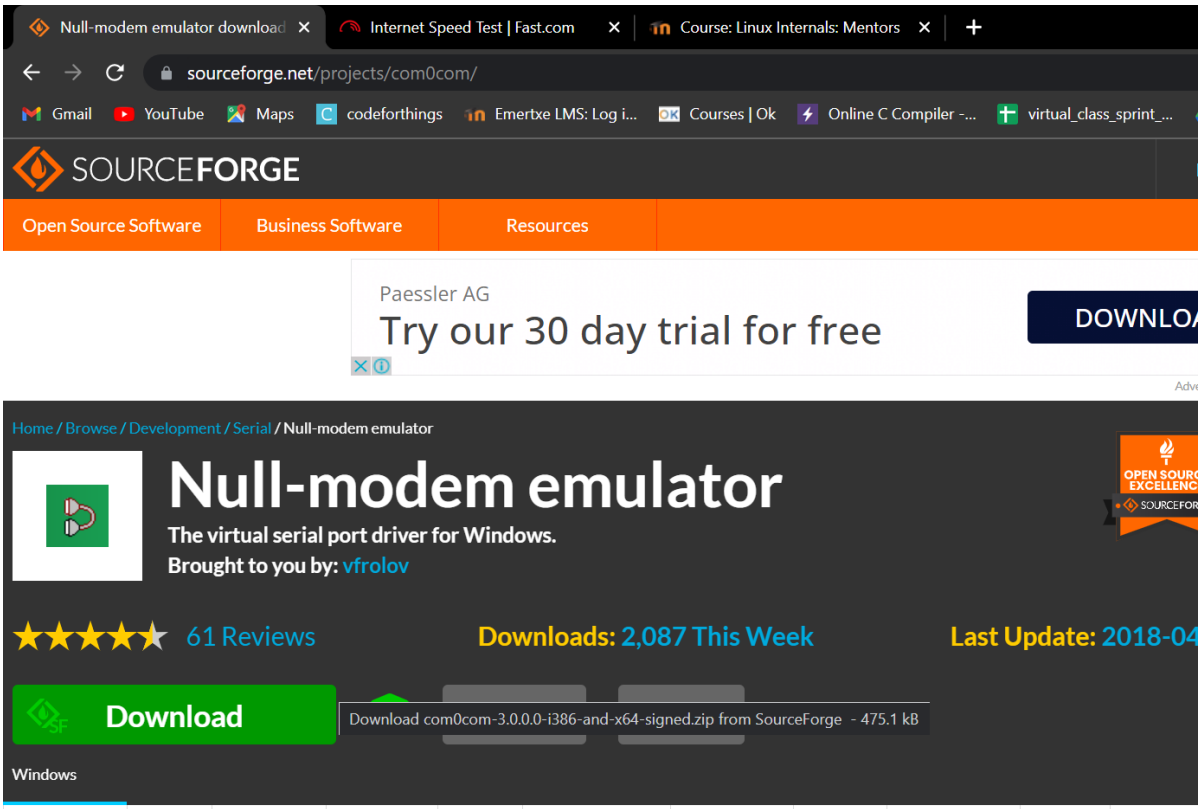
Arduino



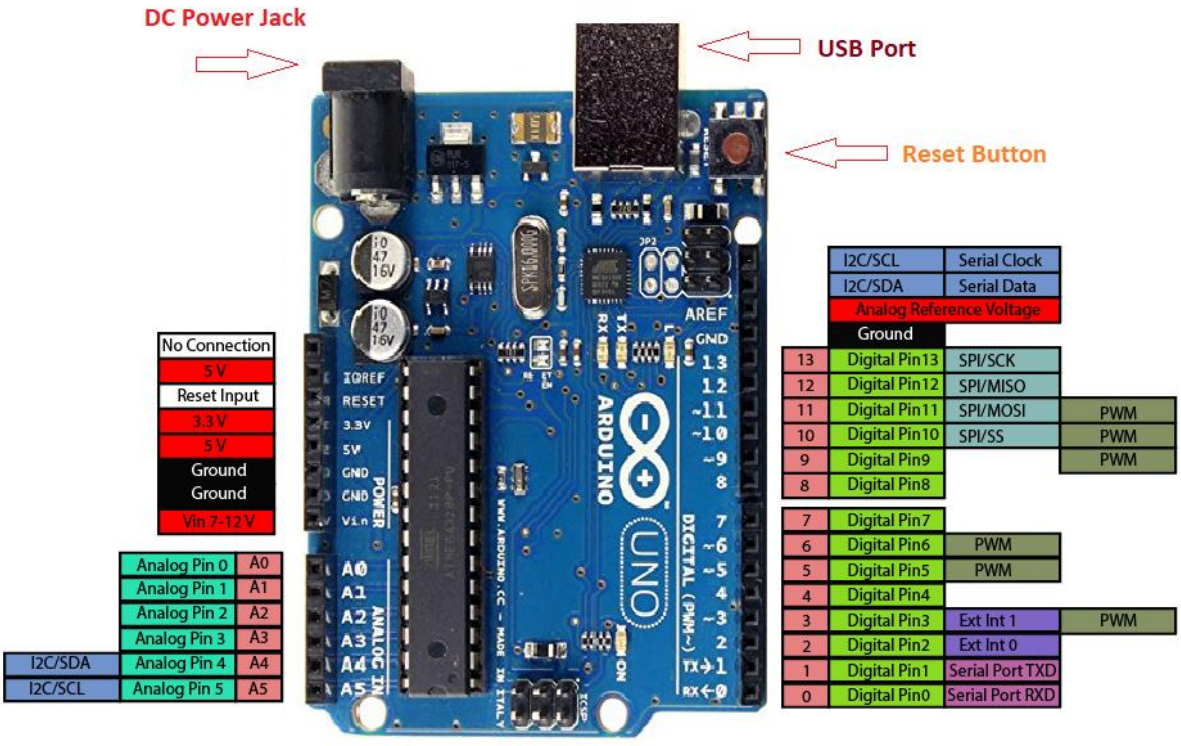
Picsimlab



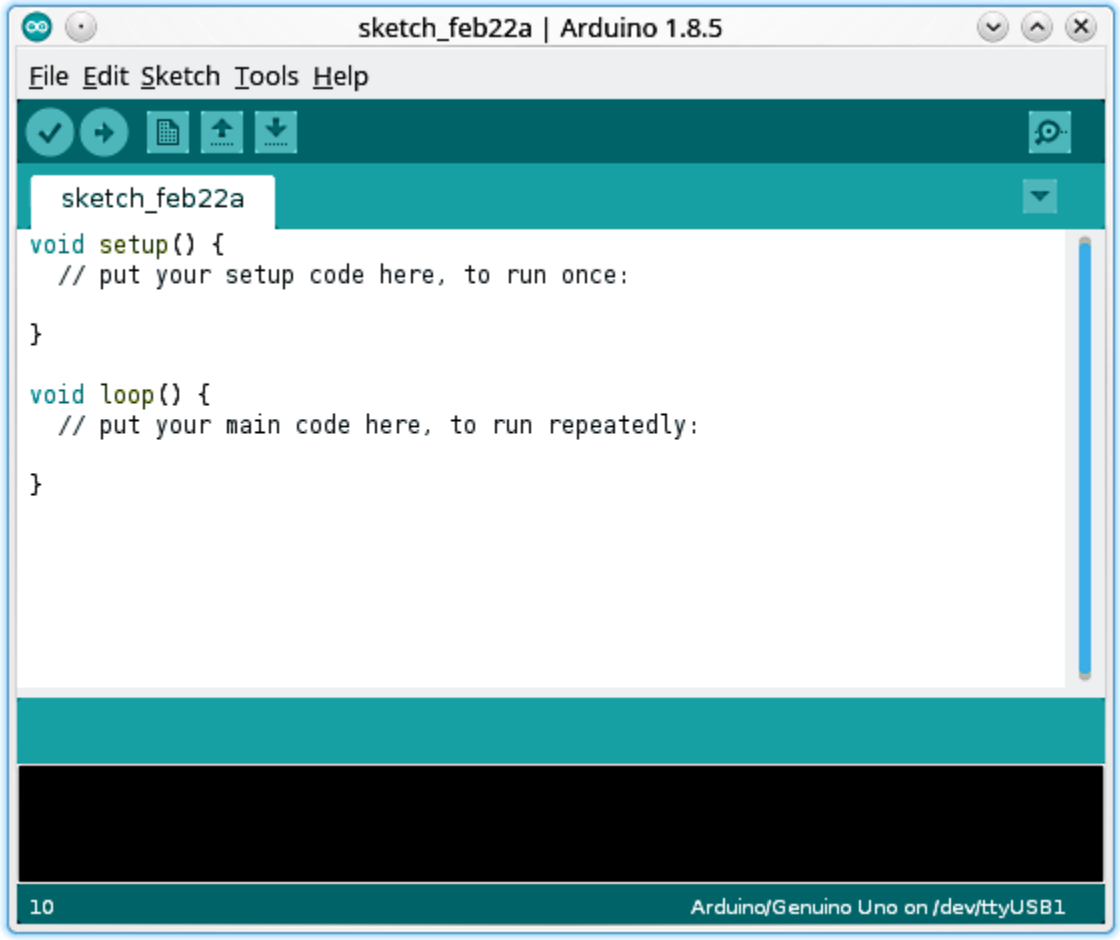
Null modem emulator



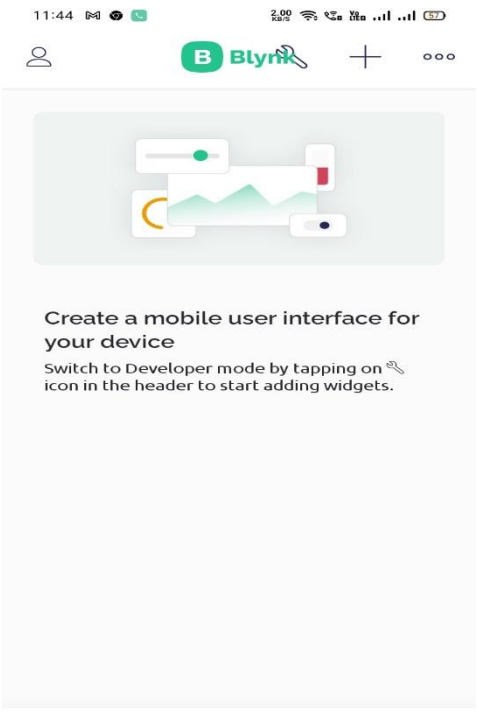
Arduino
Interface - Pinout



Arduino
IDE



Blynk



V. Project Overview: Home Automation Using Arduino and Blynk

Overview

The "Home Automation using Arduino (Picsimlab) and Blynk Application" project, developed as part of my internship, is a comprehensive simulation of a home automation system. This project leverages the Arduino platform and the Blynk IoT application to create a virtual environment where users remotely control and monitor various home devices, such as lights, temperature, and water tank levels. The entire system is simulated using the Picsimlab simulator, which emulates Arduino hardware and peripherals, making it an ideal tool for learning and experimentation without the need for physical components.

Scope of the Project

The scope of this project is to develop a simulated home automation system that is controlled and monitored remotely using the Blynk mobile application. The system includes the following functionalities:

- Light Control: Users can turn lights on and off remotely using the Blynk app. The system simulates the behavior of LEDs to represent the lights in the home.
- Temperature Monitoring and Control: The system simulates a temperature sensor, allowing users to monitor the home temperature and adjust it as needed through the Blynk app.
- Water Tank Management: The system simulates the inflow and outflow of water in a water tank, providing real-time monitoring and control via the Blynk app.

All functionalities are implemented using the Picsimlab simulator, which emulates Arduino hardware and peripherals. The Blynk app serves as the primary interface for controlling and monitoring the system.

Assumptions

The following assumptions were made during the development of this project:

- Simulated Environment: All components, including LEDs, temperature sensors, and water tank systems, are simulated using the Picsimlab simulator. No physical hardware is used.
- Blynk IoT Application: The Blynk mobile application is used as the primary interface for controlling and monitoring the system. Users have access to the Blynk app and are familiar with its basic functionalities.
- Arduino Compatibility: The Picsimlab simulator is fully compatible with the Arduino programming language and libraries used in the project.
- Internet Connectivity: The Blynk app requires an active internet connection to communicate with the simulated Arduino system. Users have access to a stable internet connection.

Dependencies

The successful implementation of this project depends on the following:

- Blynk IoT Platform: The project relies on the Blynk IoT platform for remote control and monitoring. The Blynk app provides the necessary interface for users to interact with the system.
- Picsimlab Simulator: The entire project is developed and tested using the Picsimlab simulator, which emulates Arduino hardware and peripherals.
- Arduino Libraries: The project depends on specific Arduino libraries, such as the LiquidCrystal_I2C library for display interfacing and the Blynk library for IoT connectivity.
- Programming Knowledge: A basic understanding of Arduino programming (C/C++) and familiarity with concepts such as digital I/O, analog input, and serial communication .

Constraints

The following constraints affected the development and implementation of the project:

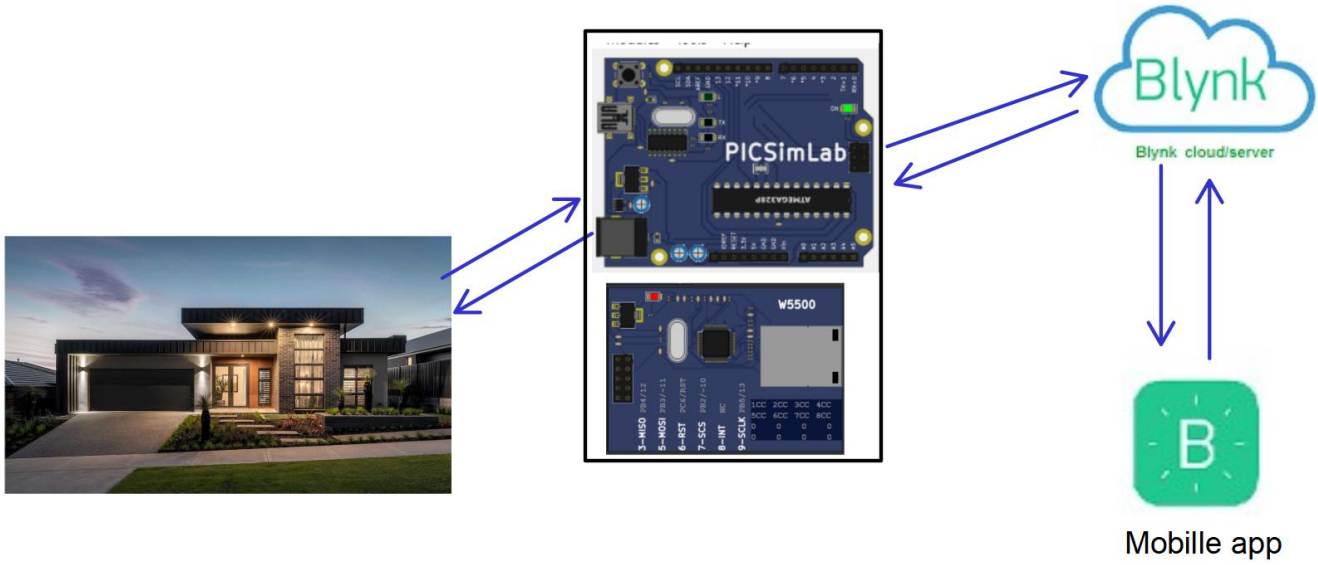
- Simulation Limitations: Since the project is entirely simulated, there are limitations in replicating real-world behaviors of hardware components. For example, the response time of simulated sensors may differ from physical sensors.
- Blynk App Limitations: The functionality of the Blynk app is limited to the features provided by the platform. Customizations beyond the app's capabilities are not possible.
- Resource Availability: The project assumes that students have access to the necessary software tools, including the Picsimlab simulator, Arduino IDE, and Blynk app. Lack of access to these tools may hinder project development.
- Internet Dependency: The Blynk app requires an internet connection for remote control and monitoring. Any disruption in internet connectivity may affect the functionality of the system.

VI. Implementation Overview

The project is implemented using the following key components and technologies:

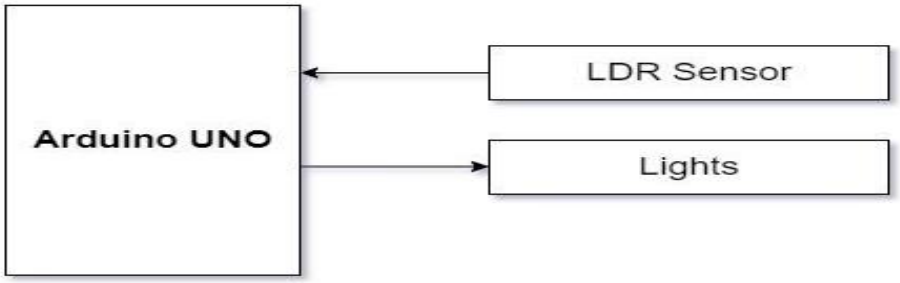
- Arduino Programming: The system is programmed using the Arduino IDE, with code written in C/C++. The program controls the simulated peripherals, such as LEDs, temperature sensors, and water tank systems.
- Picsimlab Simulator: The Picsimlab simulator is used to emulate the Arduino hardware and peripherals. This allows for the development and testing of the system without the need for physical components.
- Blynk IoT Application: The Blynk app serves as the user interface for controlling and monitoring the system. It communicates with the simulated Arduino system over the internet, allowing users to interact with the system remotely.
- Simulated Peripherals: The system simulates various peripherals, including LEDs (for light control), temperature sensors, and water tank systems. These peripherals are controlled and monitored through the Blynk app.

VII. Functional Requirements



Garden lights control

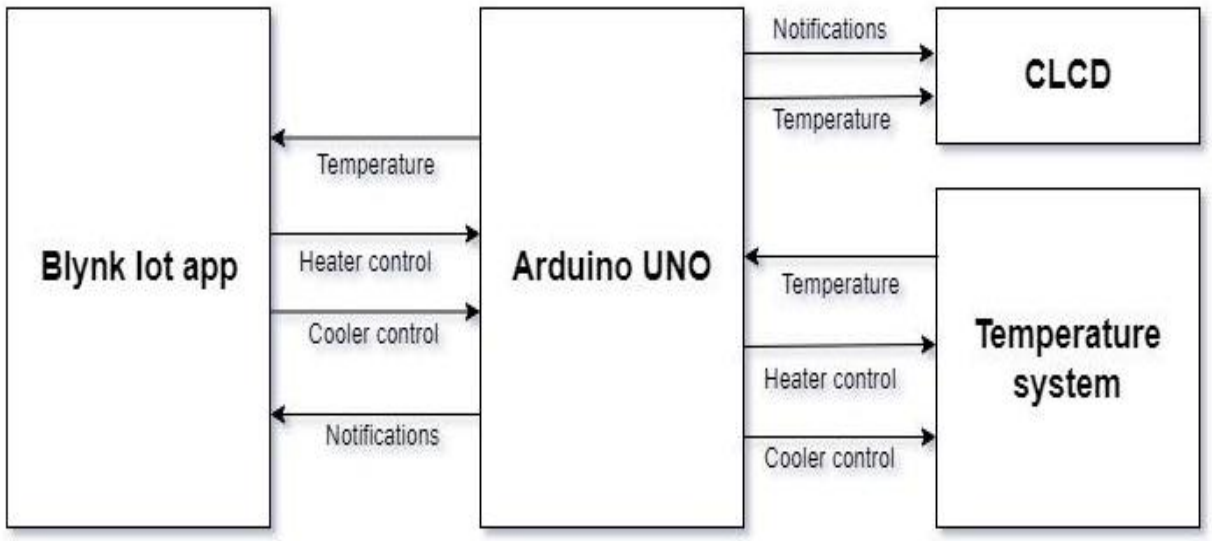
Description:
Read the LDR sensor value, based on the reading from LDR, vary the brightness of the led, which resembles controlling garden lights based on the availability of sunlight.



Requirement	1 – Garden lights control
Description	Inputs - LDR Sensor
	Process - Read the LDR sensor value, based on the reading from LDR, vary the brightness of the led, which resembles controlling garden lights based on the availability of sunlight.
	Output - Garden Lights

Temperature Control System

Description :
The temperature control system consists of a heating resistor, an LM35 temperature sensor, and a cooler. Which resembles the temperature control system at home. Read the temperature from the temperature sensor LM35 and display it on the CLCD. Control the temperature of the system by turning ON/OFF the heater and cooler through the Blynk IOT mobile app .



Requirement	2 – Temperature Control System
Description	Inputs - Temperature Sensor.
	Process - Read Temperature from temperature sensor LM35.
	Output - Display Temperature on Gauge Widget, Display Temperature on CLCD.

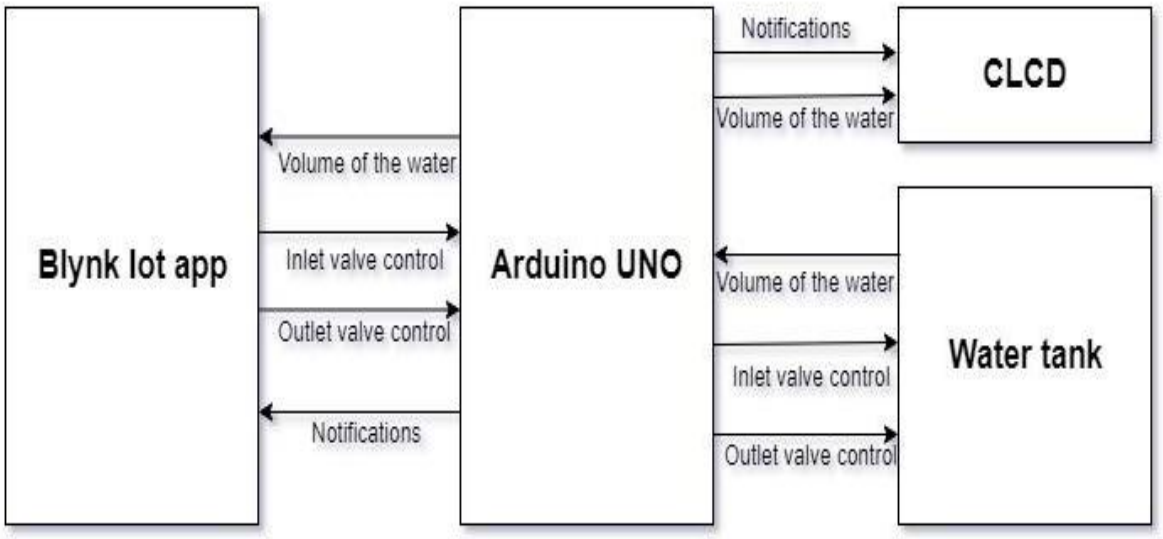
Requirement	3 – Cooler Control System
Description	Inputs - Button Widget(Cooler button) on Blynk iot app .
	Process - Detect the change in the logic level
	Output - if the Button widget is at logic high, turn ON the cooler. if the Button widget is at logic low, turn OFF the cooler.

Requirement	4 – Heater Control System
Description	Inputs - Button Widget(Heater button) on Blynk iot app .
	Process - Detect the change in the logic level
	Output - if the Button widget is at logic high, turn ON the heater. if the Button widget is at logic low, turn OFF the heater.

Requirement	5 – Threshold temperature control
Description	Inputs - Temperature sensor
	Process - Read and compare the temperature with 35 degrees
	Output - if the temperature is more than 35 turn OFF the heater and send notification to Blynk IOt app and display the same on the CLCD.

water tank inlet and outlet valve control

Description :
Read the volume of the water in the tank through Serial Communication and display it on the CLCD, control the volume of the water in the tank by controlling the inlet and outlet valve, by sending commands through serial communication. Display the volume of water in the tank on the CLCD.



Requirement	6 – Display the volume of water in the tank
Description	Inputs - Serial tank.
	Process - Read the volume of the water in the tank by sending commands through serial communication.
	Output - Display the volume of the water on Gauge Widget Display the volume of water on CLCD.

OBJ

Requirement	7 – Inlet valve control
Description	Inputs - Button Widget(Inlet valve button) on Blynk iot app .
	Process - Detect the change in the logic level
	Output - if the Button widget is at logic high, turn ON the inlet valve by sending commands through serial communication. if the Button widget is at logic low, turn OFF the inlet valve by sending commands through serial communication.

Requirement	8 – Outlet valve control
Description	Inputs - Button Widget(outlet valve button) on Blynk iot app .
	Process - Detect the change in the logic level
	Output - if the Button widget is at logic high, turn ON the outlet valve by sending commands through serial communication. if the Button widget is at logic low, turn OFF the outlet valve by sending commands through serial communication..

Requirement No	9 – Control the volume of water in the tank
Description	Inputs - Serial tank.
	Process - Read the volume of the water in the tank by sending commands through serial communication.
	Output - if the volume of water in the tank is less than 2000 ltrs turn ON the inlet valve, and send notification to the blynk mobile app and display the same on the CLCD.

VIII. User Interfaces

BLYNK Application

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

Create widgets on Mobile blynk application

Button widgets to control heater, cooler, inlet valve , outlet value.

Gauge widgets to display temperature and volume of the water in the tank on the mobile application

Terminal widgets to display the notifications whenever threshold is crossed like “Temperature is more than 35 degrees”, “” turning OFF the heater”, “water level is full “ “Water inflow disabled”

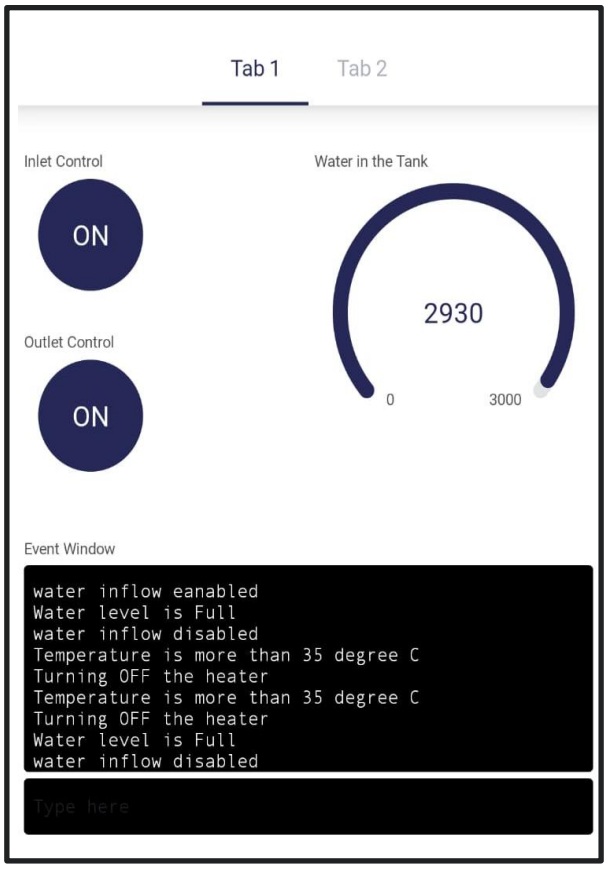


Fig 2 : (Tab1) Button widgets to control inlet valve, outlet valve, gauge to display volume of water in the tank and terminal to print the notifications.

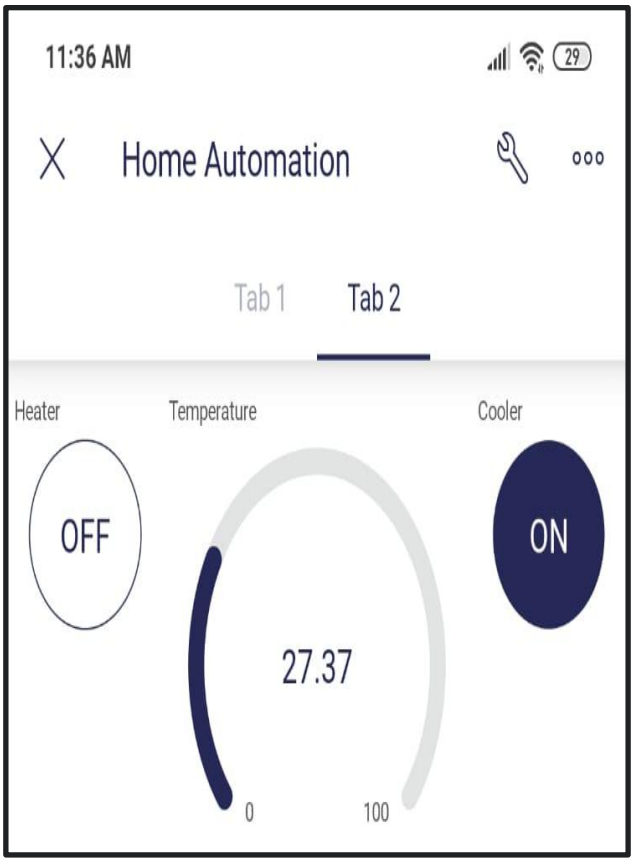


Fig 3:Tab2 Button widgets to control the heater, cooler and gauge widget to display the temperature.

Threshold control

When the heater is ON if the temperature rises above 35 degree celsius then heater should turn OFF automatically and message “Temperature is more than 35 degree celsius Turning OFF the heater” should be displayed on the virtual terminal, “HT_R OFF” on CLCD.

When the water in the tank is full, turn off the inlet valve automatically and display “water level is full water inflow disabled” .

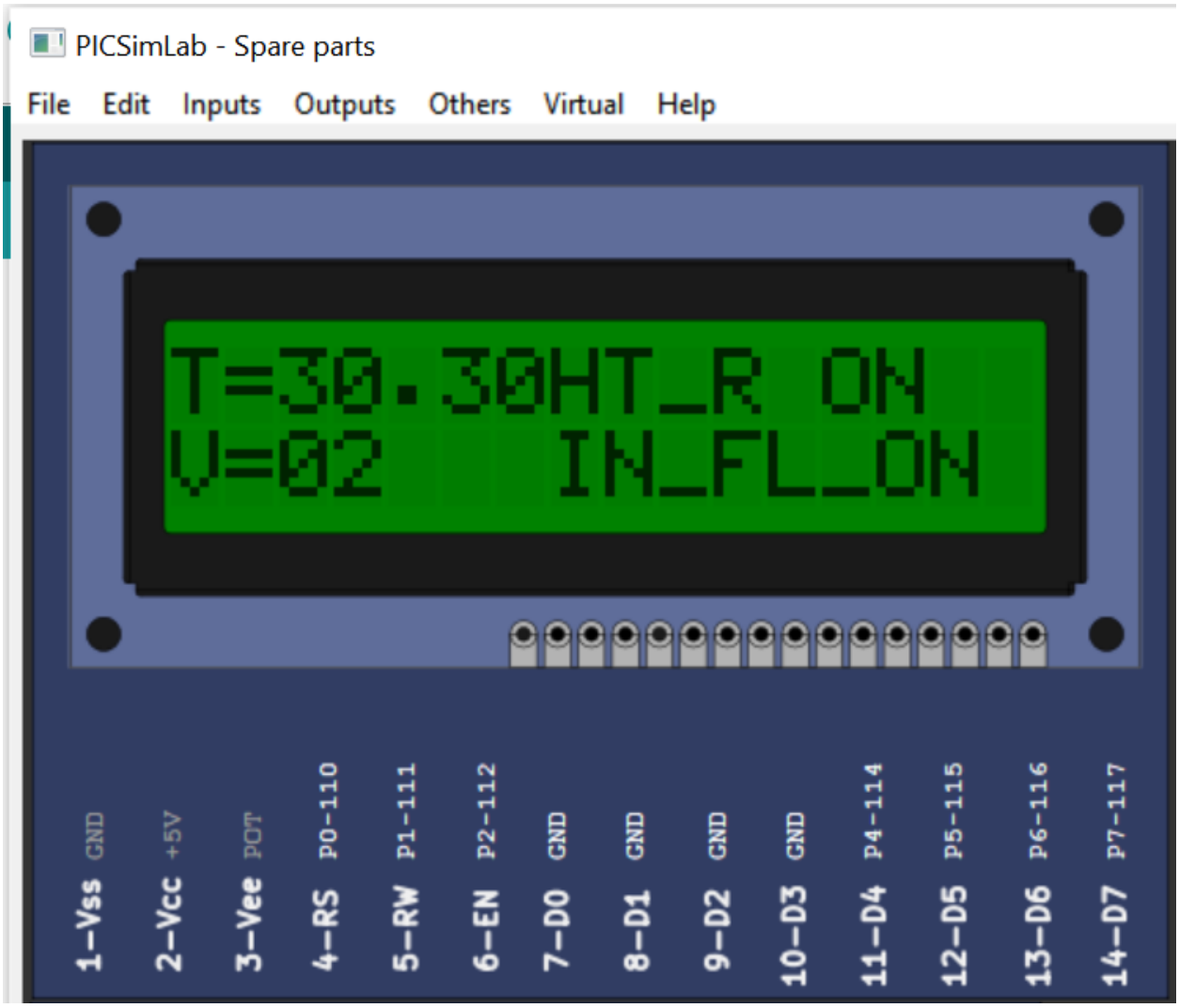


Fig 4 : Notifications on CLCD

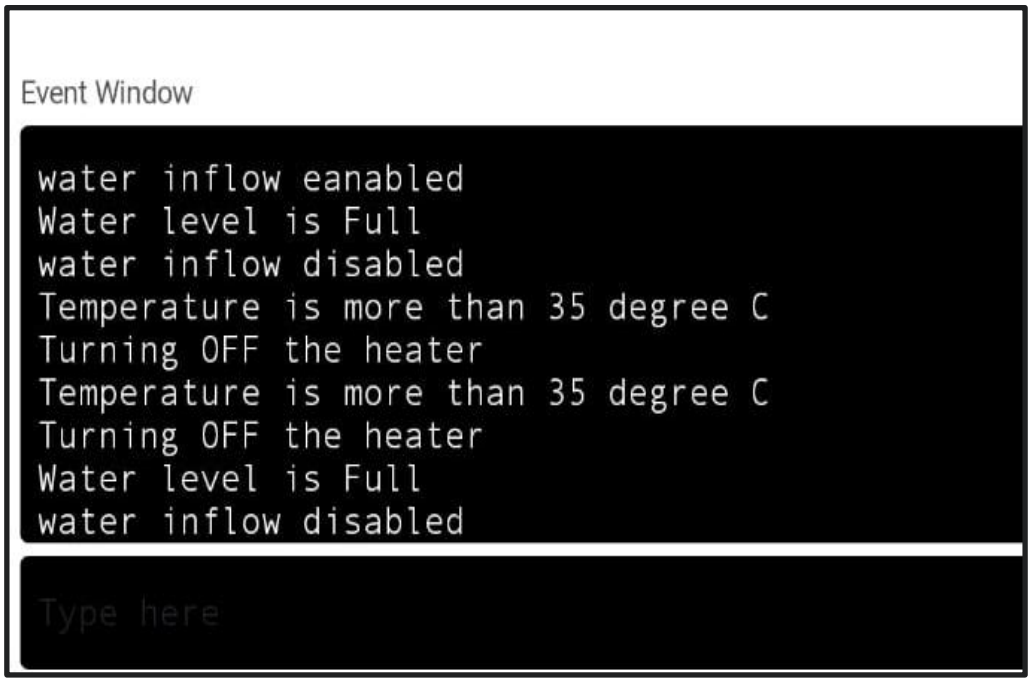


Fig 5 : Notifications on Virtual terminal

IX . Conclusion

In conclusion, the **"Home Automation using Arduino (Picsimlab) and Blynk Application"** project provides a practical and hands-on learning experience in embedded systems and IoT. By simulating a home automation system, the project demonstrates how Arduino and IoT technologies can be used to control and monitor home devices remotely. The use of the Picsimlab simulator allows for the development and testing of the system without the need for physical hardware, making it an ideal tool for learning and experimentation.