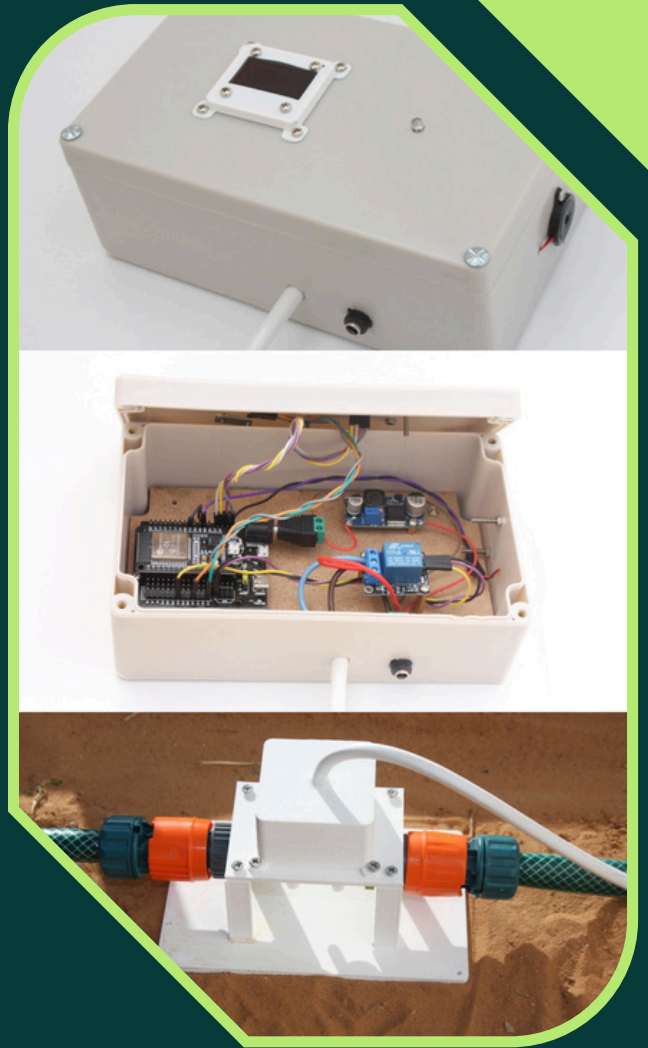


APRIL 2024 - JUNE 2024

Smart Watering System

Internet of Things Project



Presented By: **Mohamed Adam Jemal**

Computer engineering student | Internet of Things and Embedded Systems

Table of Contents

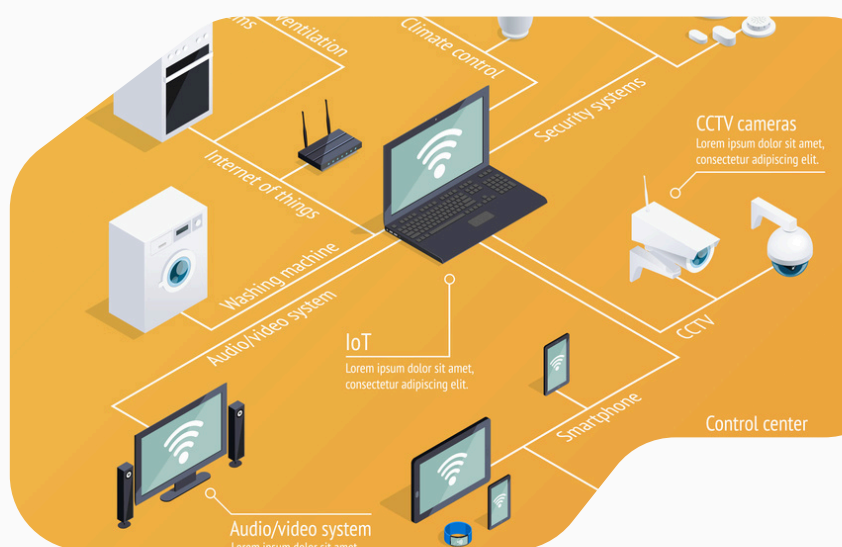
1. What is IoT ?	3
2. Project Introduction	4
3. Objectives	5
4. Methodology	6
4.1. Hardware Components	6
4.2. Software Components	7
4.3. Implementation Steps	8
5. Results	9
5.1. System Functionality	9
5.2. Performance Evaluation	10
6. Conclusion	11
7. Code Source	12
8. 3D Models	13
9. Gallery	14/15
10. Video Link	16

What is IoT?

The Internet of Things (IoT) refers to the network of physical objects—"things"—embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet.

These objects can range from ordinary household items to sophisticated industrial tools.

IoT enables remote monitoring and control of these connected devices from anywhere the internet is available, leading to improved efficiency, accuracy, and economic benefit by reducing human intervention.



Introduction

This project implements an IoT application for a smart watering system using the ESP32 board and Arduino IoT Cloud. It enables the remote control of a water valve. Also, it offers real-time monitoring of temperature and humidity values, which are displayed on an OLED screen, and integrates with the Arduino IoT Cloud for real-time data visualization.

This report provides a structured explanation of the Smart Watering System project. In addition, it represents how IoT principles enable efficient interconnectivity of devices.

Objectives

The main objectives of this project are:

- To develop a smart watering system using the ESP32 microcontroller and Arduino IoT Cloud.
- To enable remote monitoring and control of the watering system from anywhere in the world where the internet is available.
- To integrate time and weather APIs for real-time monitoring of temperature and humidity.
- To implement automatic shutdown features to prevent accidental overuse.

Methodology

Hardware Components

The system utilizes the following hardware components:

- ESP32 microcontroller board for system control and connectivity.
- Voltage regulator to power the ESP32 board safely.
- 12V electrical valve controlled via a relay.
- 1.3-inch OLED screen for displaying system status and environmental data.
- RGB LED and passive buzzer for visual and auditory feedback.

Methodology

Software Components

- Arduino IDE and libraries for programming the ESP32 include:
 - U8g2
 - HTTPClient
 - ArduinoLoTCloud
 - Arduino_ConnectionHandler
 - ArduinoJson
- Arduino IoT Cloud for remote data visualization and control.
- External APIs (Open Meteo and timeapi.io) for obtaining weather and time information.

Methodology

Implementation Steps

1. Hardware Setup:

- Connecting the ESP32 to the voltage regulator, relay, the OLED screen, the passive buzzer and the RGB LED.

2. Software Development:

- Developing software to control the valve, which connects to the internet via WiFi, in order to fetch data through APIs and communicate with the Arduino IoT Cloud.
- Implement features such as automatic shutdown after one hour of operation and status indicators using a buzzer, an RGB LED, and an OLED screen.

Results

System Functionality

- The system successfully enables remote monitoring and control of the watering system via the Arduino IoT Cloud.
- Real-time temperature and humidity data are displayed on the OLED screen and synchronized with the Arduino IoT Cloud dashboard.
- The RGB LED and passive buzzer provide intuitive system status indications and alerts for users.

Results

Performance Evaluation

- The system operates reliably under normal conditions, maintaining accurate data that is fetched through APIs, and offering a seamless, responsive remote control.
- The system status, time, date, and real-time temperature and humidity data are displayed on the OLED screen.
- Automated shutdown features effectively prevent overuse of the watering system, enhancing operational safety and efficiency.

Conclusion

In conclusion, this project successfully demonstrates the implementation of an IoT application using the ESP32 board and Arduino IoT Cloud for a smart watering system.

It highlights the remote control capabilities of the water valve, real-time monitoring of temperature and humidity via an OLED screen, and seamless integration with Arduino IoT Cloud for data visualization.

Through the application of IoT principles, the project enhances water resource management by eliminating the need for physical presence to open and close the valve, thereby overcoming challenges related to distance and time.

This showcases the potential of IoT in improving agricultural practices and resource efficiency.



Code Source

Github Link

https://github.com/Mohamed-Adam-Jemal/Smart-Watering-System/blob/main/Software/watering_system_software/watering_system_software.ino

Drive Link

https://drive.google.com/file/d/1T4hvSLwMfM5LH_atdQyDVIXmYS1sz4vS/view?usp=drive_link

```

1 #include <Ug2lib.h>
2 #include <HttpClient.h>
3 #include <ArduinoJson.h>
4 #include <ArduinoIoCloud.h>
5 #include <Arduino_ConnectionHandler.h>
6
7 #define valve_pin 33
8 #define buzzer_pin 17
9 #define green_led_pin 13
10 #define red_led_pin 27
11
12 USX8_SH1106_128X64_NONAME_HW_I2C u8x8(USX8_PIN_NONE); //OLED Screen Initialization
13
14
15 const char SSID[] = ""; //Your Network SSID (name)
16 const char PASS[] = ""; //Your Network password
17 const char DEVICE_LOGIN_NAME[] = ""; //Your Device ID (Given by the Arduino IoT Cloud Platform)
18 const char DEVICE_KEY[] = ""; //Your Device Secret Key (Given by the Arduino IoT Cloud Platform)
19
20 WiFiConnectionHandler ArduinoIoPPreferredConnection(SSID, PASS);
21
22 unsigned long lastWeatherUpdate = 0; // Variable to store the last weather update time
23 unsigned long lastTimeUpdate = 0; // Variable to store the last time update time
24 unsigned long valveStartTime = 0; // Variable to store the time when the valve was opened
25 bool valveOpen = false; // Flag to indicate if the valve is currently open
26 const unsigned long valveMaxTime = 3600000; //The valve can be opened for a maximum of one hour, measured in milliseconds
27 CloudTemperature temp_gauge;
28 CloudRelativeHumidity hum_gauge;
29 bool valve_button;
30
31 void setup()
32 {
33   pinMode(valve_pin, OUTPUT);
34   pinMode(buzzer_pin, OUTPUT);
35   pinMode(green_led_pin, OUTPUT);
36   pinMode(red_led_pin, OUTPUT);
37
38   u8x8.begin();
39   u8x8.setPowerSave(0);
40   u8x8.clearDisplay();
41   u8x8.setFont(u8x8_font_chroma48medium_r);
42
43   ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
44   ArduinoCloud.setSecretKey(DEVICE_KEY);
45   ArduinoCloud.addProperty(valve_button, READWRITE, ON_CHANGE, onValveButtonClick);
46   ArduinoCloud.addProperty(hum_gauge, READ, ON_CHANGE, NULL);
47   ArduinoCloud.addProperty(temp_gauge, READ, ON_CHANGE, NULL);
48
49   ArduinoCloud.update();
50   unsigned long currentMillis = millis();
51
52   if (valveOpen) {
53     // Check if the valve has been open for more than 2 minutes
54     if (currentMillis - valveStartTime >= valveMaxTime) {
55       valveOpen = false; // Close the valve
56       valve_button = false; // Update the Flag
57       valve_button = false;
58     }
59   }
60
61   if (currentMillis - lastTimeUpdate >= 180000) {
62     // Update the last update time
63     lastTimeUpdate = currentMillis;
64     // Perform the time update
65     updateTime();
66   }
67
68   // Check if it's time to update weather
69   if (currentMillis - lastWeatherUpdate >= 600000) {
70     // Update the last update time
71     lastWeatherUpdate = currentMillis;
72     // Perform the weather update
73     updateWeather();
74   }
75
76   void initProperties()
77   {
78     ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
79     ArduinoCloud.setSecretKey(DEVICE_KEY);
80     ArduinoCloud.addProperty(valve_button, READWRITE, ON_CHANGE, onValveButtonClick);
81     ArduinoCloud.addProperty(hum_gauge, READ, ON_CHANGE, NULL);
82     ArduinoCloud.addProperty(temp_gauge, READ, ON_CHANGE, NULL);
83   }
84 }
85
86 void loop()
87 {
88   ArduinoCloud.update();
89
90   unsigned long currentMillis = millis();
91
92   if (valveOpen) {
93     // Check if the valve has been open for more than 2 minutes
94     if (currentMillis - valveStartTime >= valveMaxTime) {
95       valveOpen = false; // Close the valve
96       valve_button = false; // Update the Flag
97       valve_button = false;
98     }
99   }
100
101   if (currentMillis - lastTimeUpdate >= 180000) {
102     // Update the last update time
103     lastTimeUpdate = currentMillis;
104     // Perform the time update
105     updateTime();
106   }
107
108   // Check if it's time to update weather
109   if (currentMillis - lastWeatherUpdate >= 600000) {
110     // Update the last update time
111     lastWeatherUpdate = currentMillis;
112     // Perform the weather update
113     updateWeather();
114   }
115
116   void initProperties()
117   {
118     ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
119     ArduinoCloud.setSecretKey(DEVICE_KEY);
120     ArduinoCloud.addProperty(valve_button, READWRITE, ON_CHANGE, onValveButtonClick);
121     ArduinoCloud.addProperty(hum_gauge, READ, ON_CHANGE, NULL);
122     ArduinoCloud.addProperty(temp_gauge, READ, ON_CHANGE, NULL);
123   }
124 }

```

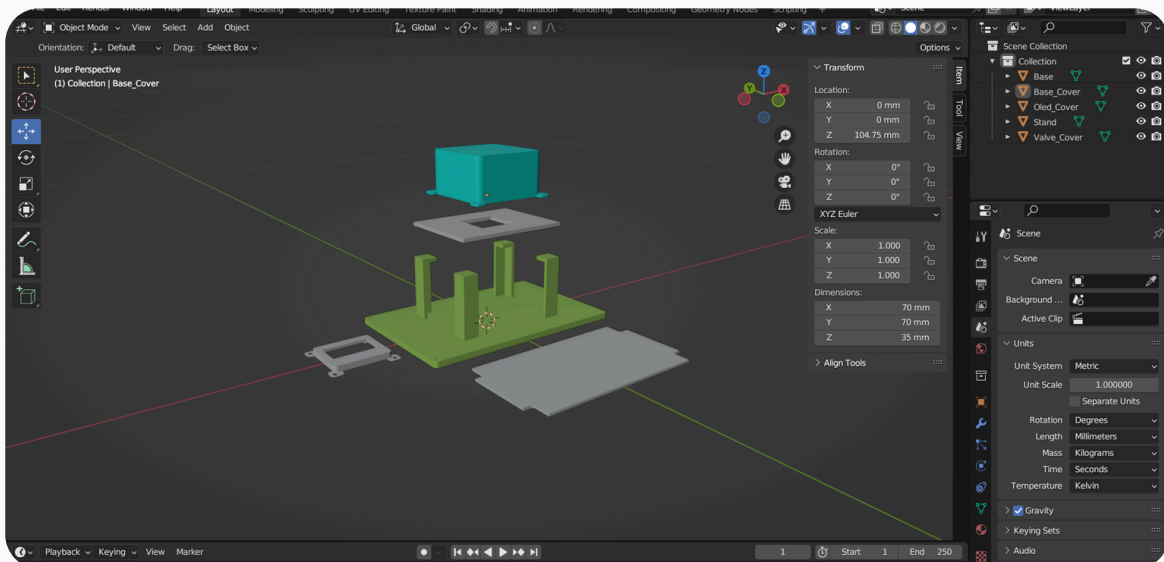
3D Models

Github Link

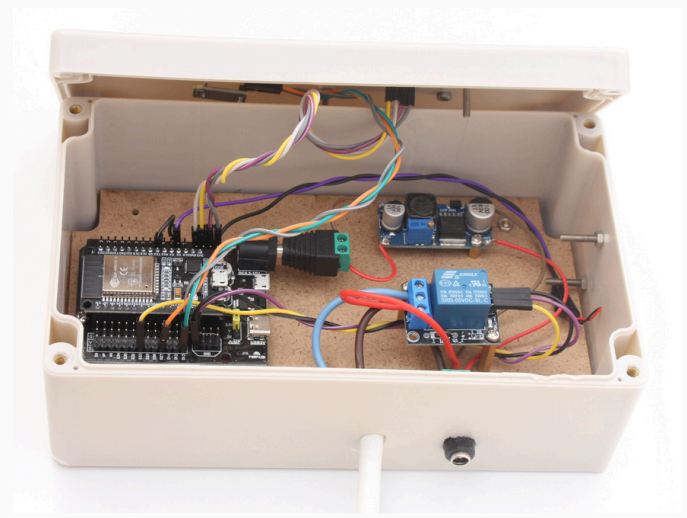
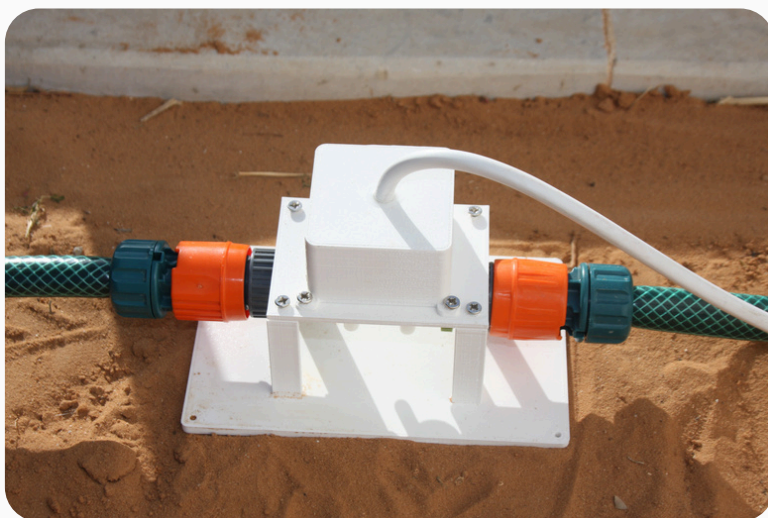
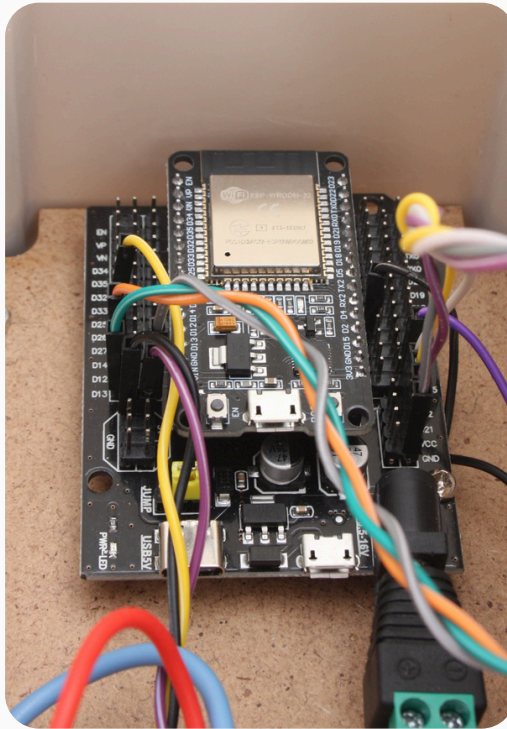
<https://github.com/Mohamed-Adam-Jemal/Smart-Watering-System/tree/main/3D%20Models>

Drive Link

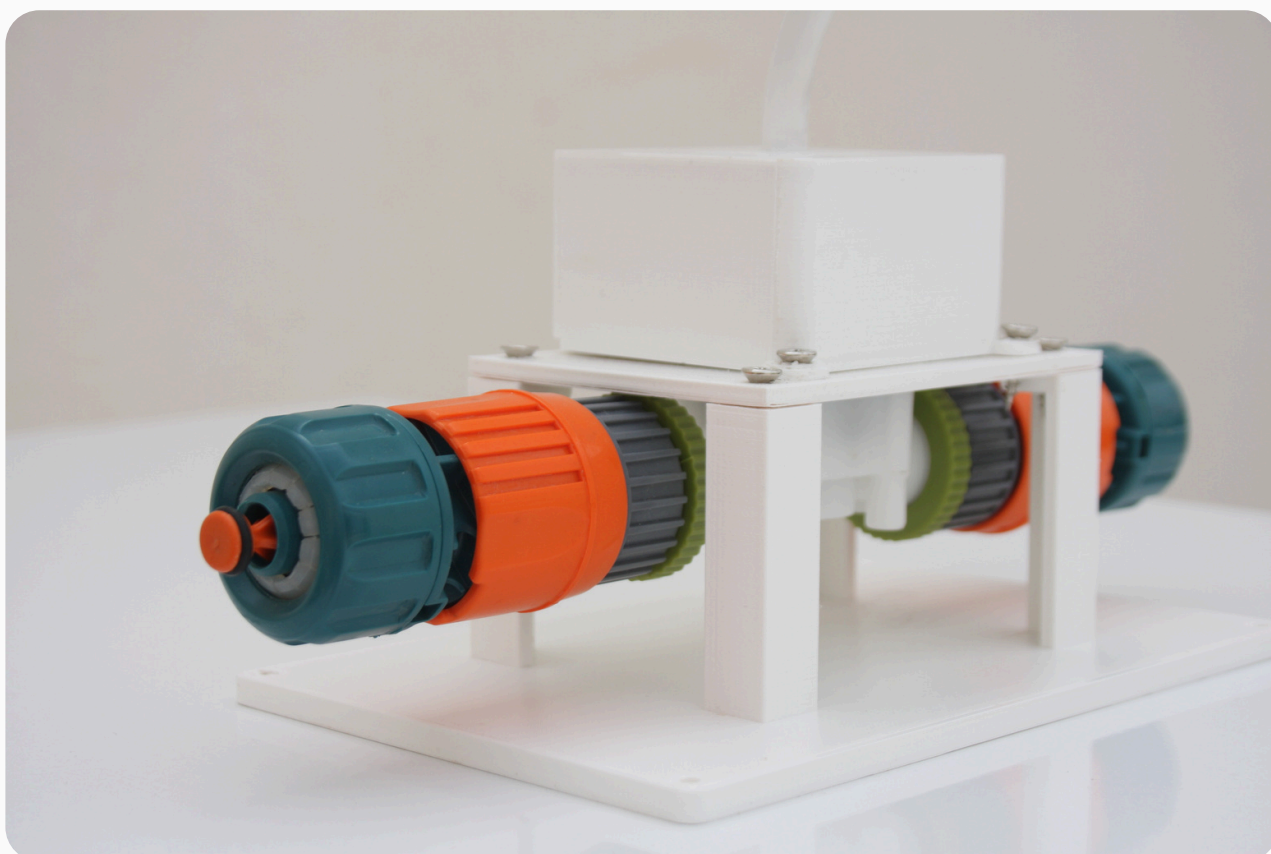
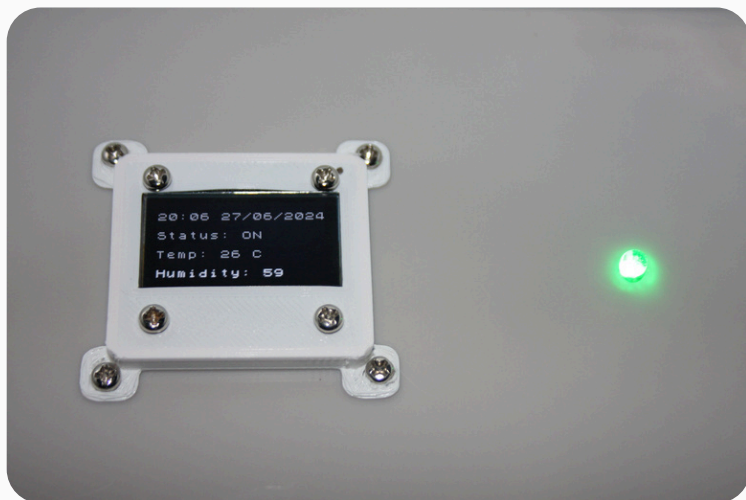
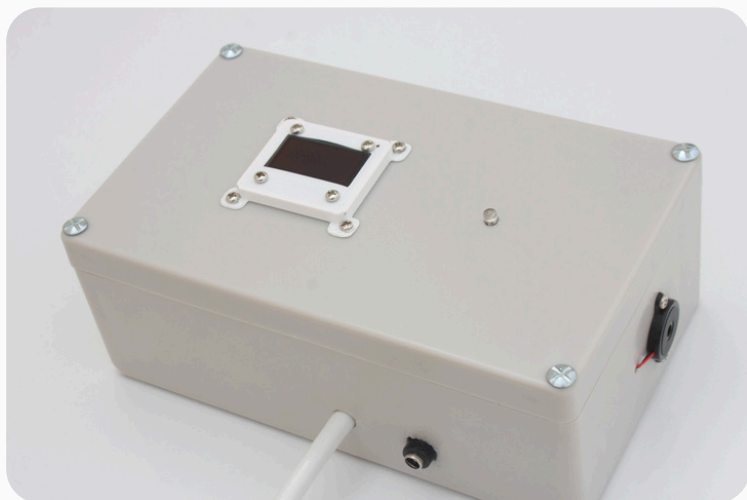
https://drive.google.com/drive/u/0/folders/1jcyWkrPCs5DikDCS6Qa_xS-h-9EvUMFR



Gallery



Gallery



Video



**[HTTPS://DRIVE.GOOGLE.COM/FILE/D/15KWSG3-
FLIUXG0EGQTEK3NPXRO4TNPB5/VIEW?
USP=DRIVE_LINK](https://drive.google.com/file/d/15KWSG3-FLIUXG0EGQTEK3NPXRO4TNPB5/view?usp=drive_link)**



Contact me for further details

- mohamed.adam.jemal@gmail.com
- github.com/Mohamed-Adam-Jemal
- linkedin.com/in/Mohamed-Adam-Jemal