Smart Home Automation & Garden Management System

A Tentative report on the Industrial Internship Project.

This project showcases an IOT based Smart Home Automation and Garden Management System built with an Arduino Nano, ESP32, and a variety of sensors. The Nano measures soil temperature and moisture, automatically controls a water pump based on moisture levels, and transmits soil data to the ESP32 via XBee S2C. The ESP32 includes a PIR motion sensor, LDR light sensor, DHT11 temperature and humidity sensor, MQ135 air quality sensor, and MQ2 gas sensor. When the MQ2 detects smoke, it activates a buzzer and regulates room illumination using PIR sensor motion detection. All sensor data is transmitted to Adafruit IO for real-time visualization and control. The dashboard allows the user to turn on and off a fan, a light, and a water pump, and it also incorporates an "Intruder Mode" switch that activates the alarm when smoke or motion is detected, boosting security and automation.

Smart Home Automation & Garden Management System

A Tentative report on the Industrial Internship Project.

CONTENTS

Components Utilized

The Components used in this project are: -

- Arduino Nano
- ESP-32 Devkit V1
- Xbee S2C Modules with Generic USB Adapter
- DHT 11 Sensor
- HC-SR501 PIR Motion Sensor
- LDR Light Sensor
- MQ2 Gas Sensor
- MQ135 Air Quality Sensor
- Soil Moisture Sensor
- PT100 Soil Temperature Sensor
- 128*32 OLED Display
- 5V Single Channel Relay Module
- Adafruit IO

Component Description

Arduino Nano

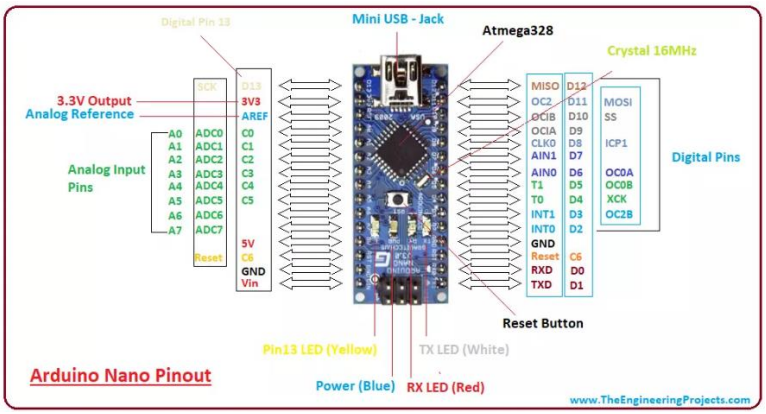

Fig. 1 Arduino NANO Diagram with Pinout [1]

The Arduino Nano is a small, breadboard-friendly microcontroller based on the ATmega328p, created by Arduino.cc in 2008. It has 14 digital pins, 8 analogue pins, 2 reset pins, and 6 power pins, and operates at 5V with an input voltage range of 7-12V. The Nano can handle a maximum current of 40mA per pin and features a 16 MHz crystal oscillator. It is programmed using the Arduino IDE and connects to computers via a Type-B Micro USB port. With 16KB or 32KB flash memory, 2KB SRAM, and 1KB EEPROM, the Nano is perfect for applications that require a tiny, flexible microcontroller without a DC power connector.

ESP-32 Devkit V1

The DoIt ESP32 DevKit V1 is a development board that includes Espressif's ESP32 microcontroller. It is intended for Internet of Things (IoT) projects and applications. The board supports Wi-Fi and Bluetooth, as well as a number of input/output interfaces such as GPIO, analogue inputs, UART, SPI, and I2C. It also contains a standard form size that allows for easy integration into projects, as well as a USB connector for power and programming. The ESP32 DevKit V1 is popular among hobbyists and educators due to its ease of use and adaptability in a wide range of electronics projects.

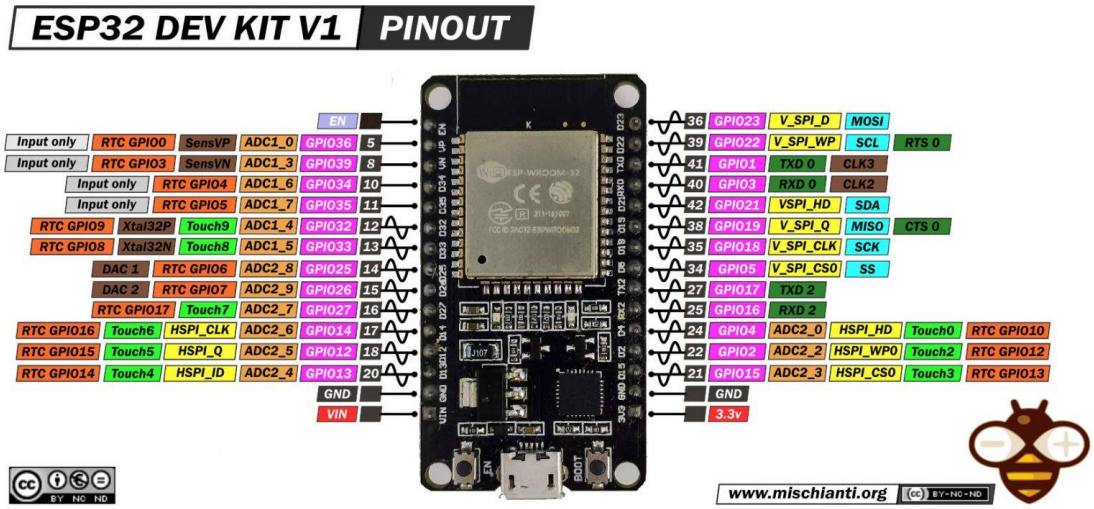

Fig. 2 ESP-32 Devkit V1 [2]

XBee S2C Modules with Generic USB Adapter

The XBee S2C module is a multifunctional ZigBee RF module designed to provide dependable and robust wireless communication. It uses the ZigBee PRO protocol, making it excellent for constructing mesh networks. The S2C module has S2C hardware that improves performance, reduces power consumption, and increases RF sensitivity compared to prior generations. It has a range of up to 1200 metres (outside line of sight) and a data rate of 250 kbps. The module is compatible with XBee's large adapter board lineup and can be simply configured using Digi's XCTU software. It is appropriate for a wide range of applications, including home automation, industrial control, and wireless sensor networks.

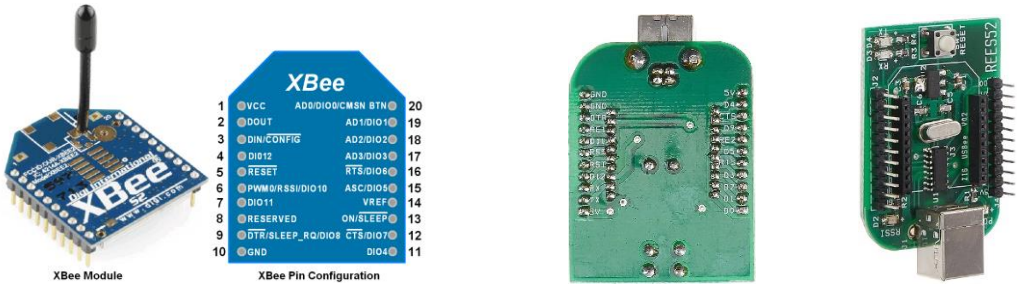

Fig. 3 Xbee S2C Module [4] & Generic USB Adapter for Xbee Module [3]

The Generic XBee ZigBee Adapter Board with USB Interface is a user-friendly USB-to-serial base unit that works with all XBee modules, including Series 1 and Series 2.5, in both basic and Pro configurations. Simply insert the XBee module into the adapter, add a USB cable, and you'll have immediate access to the serial and programming ports. The board has a 3.3V regulated

power source and separates all XBee pins for easier prototyping. Furthermore, it has a reset button and a small footprint, making it an excellent tool for configuring, programming, and working with XBee modules.

DHT-11 Sensor

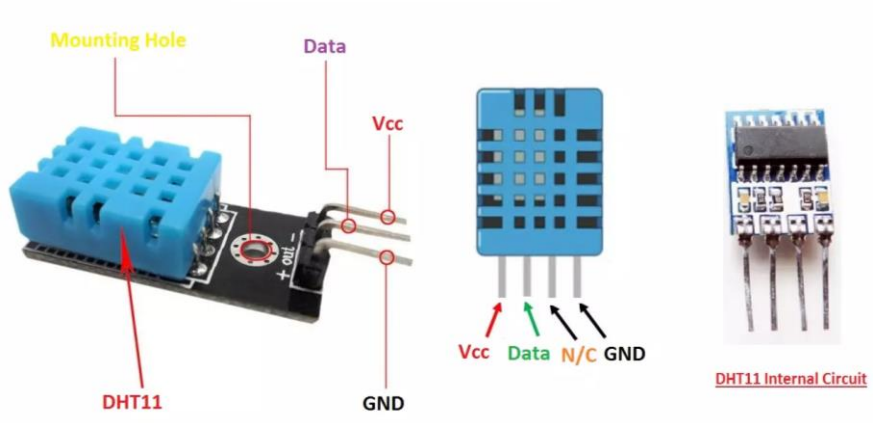

Fig. 4 DHT11 Sensor with Pinout [5]

The DHT11 is a low-cost digital sensor designed to measure temperature and humidity. It uses a capacitive humidity sensor and a thermistor to monitor the ambient air, resulting in reliable and consistent readings. The sensor generates digital signals and has an accuracy of ±2°C for temperature (0-50°C) and ±5% for humidity (20-90% RH). It is simple to interact with microcontrollers like Arduino, which use a single digital pin for data exchange. The DHT11 is ideal for basic weather stations, home automation systems, and other applications that require environmental data monitoring.

HC-SR501 PIR Motion Sensor

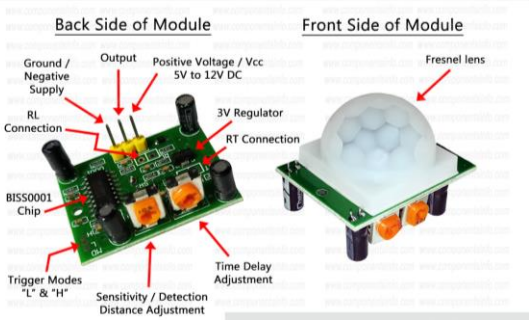

Fig. 5 HC SR501 PIR Motion Sensor [6]

The HC-SR501 is a passive infrared (PIR) motion sensor that detects human movement. It detects infrared radiation from human bodies within a detection range of approximately 7 metres and has a delay time of 5 seconds to 5 minutes. When motion is detected, the sensor sends a digital high signal, and when there is no motion, it sends a low signal. It has configurable sensitivity and time delay settings, making it perfect for use with automatic lighting, security systems, and motion-activated devices. The HC-SR501 is straightforward to integrate with microcontrollers and requires a 5V power source.

LDR Light Sensor

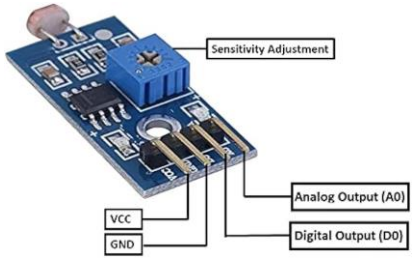

Fig. 6 LDR Light Sensor

An LDR (Light Dependent Resistor) light sensor module measures light intensity. It employs a photoresistor, whose resistance lowers as ambient light increases. The module translates the resistance change into an analogue signal that can be read by microcontrollers such as Arduino. It runs on a 3.3V-5V power source and is often used in applications including automatic lighting, light level detection, and light-activated switches. The LDR module is easy to operate, making it excellent for projects requiring light detection.

MQ2 Gas Sensor

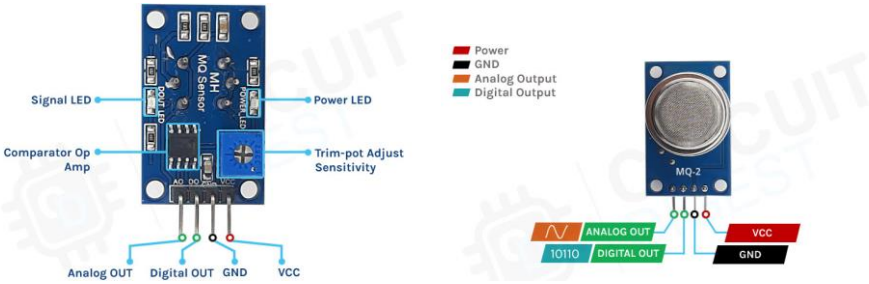

Fig. 7 MQ2 Gas Sensor [8]

The MQ2 sensor is an air quality sensor that detects methane, propane, carbon monoxide, and smoke. It includes a sensitive ceramic element that adjusts resistance in response to gas

concentrations. Microcontrollers can read the sensor's analogue signal, which is proportional to gas concentration. It is widely utilised in applications such as gas leak detection, smoke alarms, and air quality monitoring. The MQ2 uses a 5V power supply and requires a warm-up period to produce consistent readings.

MQ135 Air Quality Sensor

Fig. 8 MQ135 Air Quality Sensor [9]

The MQ135 sensor can detect a variety of chemicals, including ammonia, carbon dioxide, benzene, and smoke. It has a sensitive metal oxide layer that changes resistance based on gas concentrations. The sensor generates an analogue output that fluctuates with gas levels and can be read by microcontrollers. It is widely utilized in air quality monitoring and indoor pollution detection. The MQ135 runs on a 5V power supply and requires a warm-up period to achieve consistent readings.

Soil Moisture Sensor

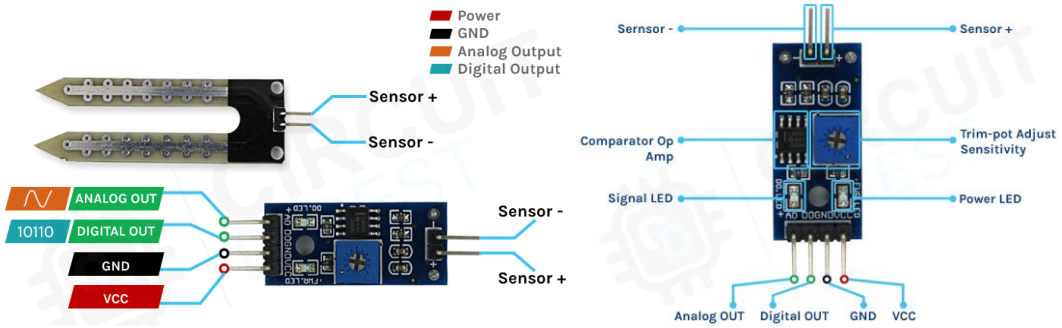

Fig. 9 Soil Moisture Sensor [10]

A soil moisture sensor monitors the water content of the soil. It typically consists of two probes that detect the soil's resistance or capacitance; resistance reduces as moisture content

increases. The sensor generates an analogue or digital signal proportional to soil moisture levels. It is widely used in irrigation systems and gardening to monitor soil moisture levels and automate watering based on soil requirements. The sensor requires a 3.3V to 5V power source and is simple to interface with microcontrollers.

PT100 Soil Temperature Sensor

Fig. 10 PT100 Soil Temperature Sensor [12] with pinout [11]

The PT100 soil temperature sensor has great accuracy. It employs a platinum resistor whose resistance fluctuates with temperature; precisely, it has a resistance of 100 ohms at 0° Celsius. The sensor produces a precise and consistent output that may be read by microcontrollers or measuring systems. It is widely used in agricultural and environmental monitoring to properly measure soil temperature and promote plant growth. The PT100 sensor, which requires an appropriate interface for temperature conversion, is noted for its great precision and dependability.

*128*32 OLED Display*

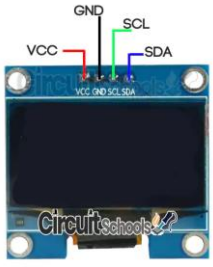

*Fig. 11 128*32 OLED Display [13]*

A 128x32 OLED display is a small, high-contrast screen with 128 columns and 32 rows of pixels. It employs organic light-emitting diodes (OLEDs) to provide bright, crisp text and images that do not require a backlight. This display provides wide viewing angles, low power consumption, and excellent visibility in a variety of lighting settings. It is widely utilised in embedded systems

and tiny electronics projects to provide crisp and sharp visual output. The display commonly communicates with microcontrollers using I2C or SPI.

5V Single Channel Relay Module

Fig. 12 5V Single Channel Relay Module [14]

A low-voltage microcontroller interfaces with a 5V single-channel relay module to control high-power devices. It has a relay switch that can handle AC or DC loads up to a predetermined current rating, usually 10A. A 5V digital signal from a microcontroller activates the module, allowing it to reliably power on and off high-voltage devices. It incorporates optocouplers that isolate and protect the microprocessor. It is commonly used in automation and control systems to make it easier to interface low-voltage electronics with high-power equipment.

Adafruit IO

Fig. 13 Adafruit IO Logo [15]

Adafruit IO is a cloud-based platform for IoT applications that includes data storage, visualization, and control. Users can collect data from sensors, display it on customizable dashboards, and issue commands to linked devices. Adafruit IO provides MQTT and REST APIs for simple integration with a variety of microcontrollers and devices. It simplifies IoT development by including data logging, triggers, and notifications, making it ideal for remote monitoring and control of IoT systems.

Methodology

Garden Management Setup

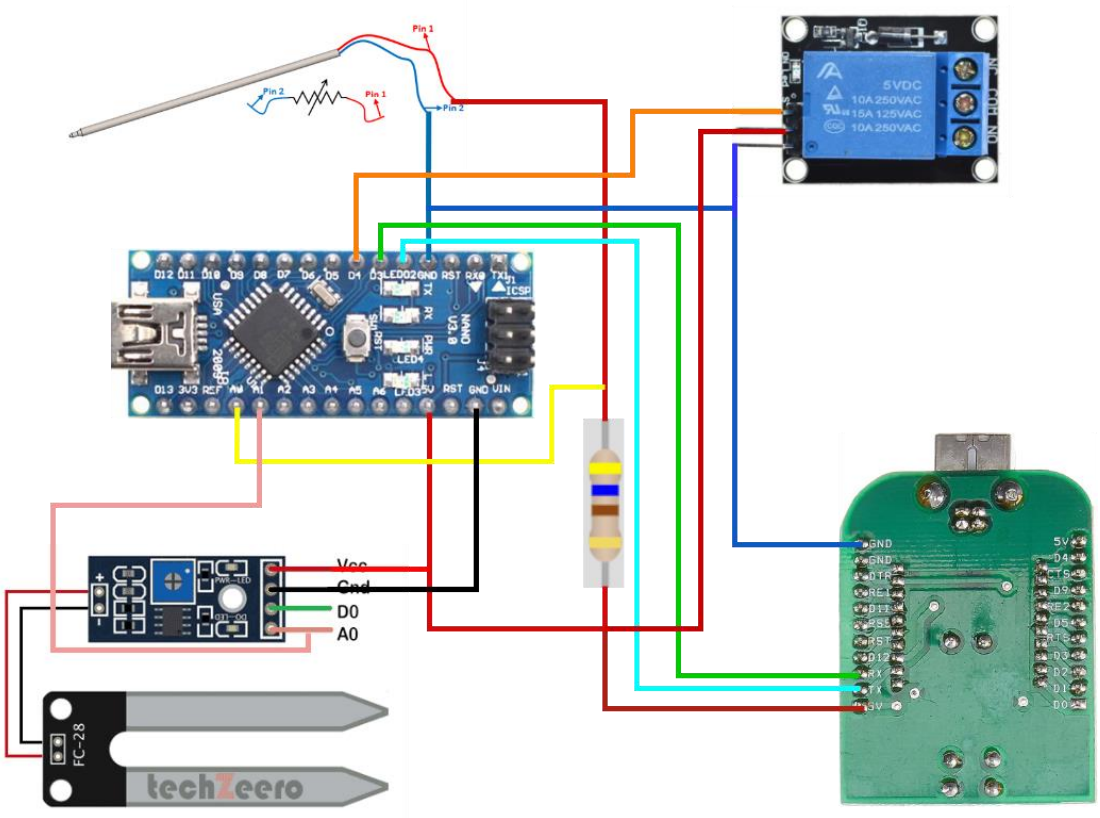

Fig. 14 Garden Management Setup

- For Garden Management System, we have connected the VCC & GND pins of the soil moisture sensor to the 5V & GND pins of NANO. We have connected the AO (Analog Output) pin of the sensor to the Analog Input pin A1 of NANO to read the analog voltage value (which should be between 0 to 1023 as the ADC of Arduino is 10 bits, i.e., any analog voltage that the analog pin receives gets converted to a 10-bit number) corresponding to the amount of moisture present in the soil and which is then mapped between 0 to 100 or in other words, we converted the received value into a percentage value for ease of reading the percentage content of moisture present in the soil. Based on the percentage of moisture content present in the soil, we have implemented a logic to turn on(or in other words will send a high signal to the data pin) a relay module (Connections of RELAY-NANO: VCC-5V, GND-GND, SIGNAL-D4), which will be connected to a water pump, in case of low moisture content (below 60%) and will turn off(will send a low signal to data pin) the relay when moisture content will go above

60%. We also define a state variable pump which is default set to false in NANO side to check the state of the pump, if it is on (variable will be set to true) or not (variable will be set to false). This variable will be checked when the moisture logic runs. So, if the pump is turned on by the logic, the state variable will be set to true & if turned off, the variable will be set to false.

- For the PT100 Soil temperature sensor, we have connected one of its ends to the GND of NANO and other end to one end of a 462 ohm resistor, then we have connected the other end of the resistor to the 5V of the NANO, and we have connected the junction between 462 ohm and PT100 to analog pin A0 of NANO. Now, the resistance across the PT100 changes when temperature around it changes & so the voltage across this voltage divider setup also changes which can be measured by A0 pin in ADC values (0 to 1023) which is converted to a voltage value by mapping it between 0 to 5V (Maximum Operating Voltage being used here in the voltage divider setup), so, we can calculate unknown resistance value of PT100 from the voltage divider setup by putting the known values in the voltage divider equation. Now knowing this value we can put it into the PT100 Temperature equation :

```
t = (r - 100.0) / 0.385
```

  where t is the soil temperature & r is the changed resistance of PT100 due to temperature.

- We have also connected the 5V and GND pins of USB Adapter of Xbee S2C module to 5V and GND pins of NANO and used SoftwareSerial header file to convert D2 and D3 pins of NANO into RX and TX pins and connected them with TX and RX pins of the USB adapter of XBee S2C respectively, so as to setup UART communication between NANO and Xbee S2C module to send sensor readings from NANO to XBee S2C and from XBee S2C to another Xbee S2C, wirelessly using Zigbee Protocol, which forwards the readings to the ESP32 connected to it, which further sends the sensor readings to Adafruit IO cloud. The ESP32 further uses the XBee S2C based wireless communication to send pump on/off commands, that are being sent from the cloud to the ESP32, to the NANO in the same way as NANO sends sensor readings to ESP32. Based on the received command, NANO turns on/off the relay module which will be connected to the water pump

Home Automation Setup

- Here, we have first connected the DHT11 sensor to ESP32, by connecting the VCC, GND & DATA pins of DHT11 to 3.3V, GND & D5 pins of ESP32 respectively. We have used the prebuilt DHT header file to take the temperature and humidity readings from the DATA pin of DHT11.

- Next, we have connected the VCC, GND & DATAOUT (DO) pins of HC SR501 PIR Motion Sensor to 3.3V, GND & D27 of ESP32 respectively. We will be digitally reading the DATAOUT pin of the PIR sensor (reading only 0 or 1 value) for motion detection, reading value: 0 for no motion & reading value: 1 for motion.

- Next, we have connected the VCC, GND & DATAOUT (DO) pins of LDR light sensor to 3.3V, GND & D35 pins of ESP32 respectively. LDR will be sending analog voltage values from its DATAOUT pin which will be converted into a 12 bit value (as ADC of ESP32 is 12 bits) and this 12 bit value will be converted into a percentage value in the same way as we did in case of soil moisture sensor previously.

- Next, we have connected the VCC & GND pins of MQ2 and MQ135 sensors to VIN & GND pins of ESP32 respectively, as these sensors need 5V at VCC for their operation. As a result, the data readings will be in 5V levels, so, connecting them directly to the data pins of ESP32 will damage the data pins as ESP32 works at 3.3V level. So, we made 2 level shifters (voltage divider circuitry) between the ANALOGOUTPUT (AO) pins of MQ135 and MQ2 and data pins of ESP32, so that the input signal in 5V level will be stepped down to 3.3V level to be safely read by the data pins of ESP32. For this level shifter circuit, we connected one end of an 1k ohm resistor to each AO of MQ2 & MQ135 and other ends of the 1K ohm resistors each to one end of a 2.16k ohm resistor and the other ends of the 2.16k ohm resistors to GND of ESP32. Then, we connected the junction of the 1K ohm resistor and 2.16K ohm resistor that is connected with the AO of MQ2 to D32 of ESP32. Also, we connected the junction of the 1K ohm resistor and 2.16k ohm resistor that is connected with the AO of MQ135 to D34 of ESP32. We then checked the air quality by using the MQ135 header file to convert the analog reading from MQ135 to CO2 content in ppm units. Also, we took the raw analog values from MQ2 and converted them into percentage value to check smoke percentage in air as we did in LDR light sensor case & we set a threshold value (50%), going above which means smoke is detected

- We then connected the VCC, GND, SDA & SCL pins of a 128*32 OLED Display to the 3.3V, GND, D21 and D22 pins of ESP32 respectively. We then used the required Adafruit libraries used for OLED to display all the sensor data (of sensors of both NANO & ESP32) on the OLED.

- We also connected VCC and GND of two relay modules to VIN and GND of ESP32 respectively. These relays will be used to turn on/off lights and fans. The SIGNAL pin for the relay for light is connected to D4 pin of the ESP32 & the SIGNAL pin for the relay for fan is connected to D2 pin of the ESP32.

- We also connected the positive & negative pins of a Buzzer to D25 & GND of ESP32 respectively. This buzzer is used as a fire alarm system in case smoke is detected in normal scenario and also as a security alarm system incase motion is detected when the esp32 will be set to intruder mode by the cloud.

- Lastly, we connected the 5V & GND pins of another Xbee S2C module to VIN & GND pins of ESP32 respectively, to setup wireless communication with NANO as mentioned previously. The RX pin of the Xbee S2C will be connected to TX2 pin of ESP32 directly as RX pin will be taking input from TX2 pin of ESP32 but as ESP32 is, the one at 3.3V level, sending signals at 3.3V level to Xbee S2C, which is at 5V level, no damage will be done as Xbee is at a higher voltage level than the received

signal level and also no high level signal is being received by TX2 pin of ESP32. But in case of TX pin of Xbee S2C, it will be sending signals at 5V level to RX2 pin of ESP32 which only takes input at 3.3V level safely. So, we will be using the same level shifter circuitry we used in case of AO pins of MQ sensors. So, we will connect the TX pin of XBee S2C to the end of 1K ohm resistor, and we will connect the RX2 pin of ESP32 to the junction of 1k ohm resistor & 2.16k ohm resistor.

- Here, our ESP32 will take sensor readings continuously and display all the sensor data (from both NANO & ESP32) at regular intervals. It will also use MQTT protocol to publish PIR motion and MQ2 smoke sensor data to Adafruit IO every 10 seconds and will publish, other sensor readings at every 15 seconds. As Adafruit IO only allows 30 messages per minute at free tier, we can't continuously take readings and send them. We will also subscribe to any messages send by the Adafruit IO cloud. Two of the messages being sent are 'ON' & 'OFF' which is used to turn ON/OFF the intruder mode in ESP32. We will define a state variable intrudermode to check if the state of esp32 is intruder mode or not. It is set default to false when the system will be powered on which means intrudermode is off. When the message received from the cloud will be 'ON', this intrudermode state will be set to true meaning intrudermode is on. In intruder mode, sensor readings will be taken continuously and published to the cloud at regular intervals normally. Just that in case of intruder mode, the alarm/buzzer will be turned on in case of motion detection and also for smoke detection, which was being turned on only for smoke detection in case of intrudermode being set to false or under normal scenarios. When the cloud will send 'off' message, this intrudermode will be set to false or intruder mode will be turned off. We also define a state variable active which will also be default set to false. This state variable will be used to determine if someone is home (if active is set to true) or not (if active is false), only if intruder mode is not on. We will also define 2 other state variables in ESP32 side, i.e., light & fan, to check state of light and fan connected to the relays of ESP32 and all these variables will be default set to false when power is supplied and both NANO & ESP32 boots up. When intruder mode is off, and motion is detected and the active state is false, then the active state will be set to true, meaning someone has entered the room, and then temperature & light intensity readings will be taken and if some certain thresholds (like high temperature & low light) are met then fans or lights or both will be turned on and state of light and fan variables will be changed to true. Again, if active mode is on and motion is detected, meaning someone left the room, then active state will go to false, and it will check the state of fan and light variables & if they are true then they are set to false and turned off. In all the above scenarios, reading of sensor data, displaying and publishing of sensor data at regular intervals keeps on happening in the background. Now apart from the 'ON', 'OFF' messages, the other messages send by the cloud are 'LIGHT', 'FAN', 'PUMP', these messages are used to check the state of state variables light, fan and pump, and reverse their state, so like, if light state is true (light is on), then on receiving 'LIGHT' message, the light state will be reversed to false and light will be turned off and vice versa, like if it was off then on receiving the message it will turn on and these messages work in the same way for fan & pump variables too.

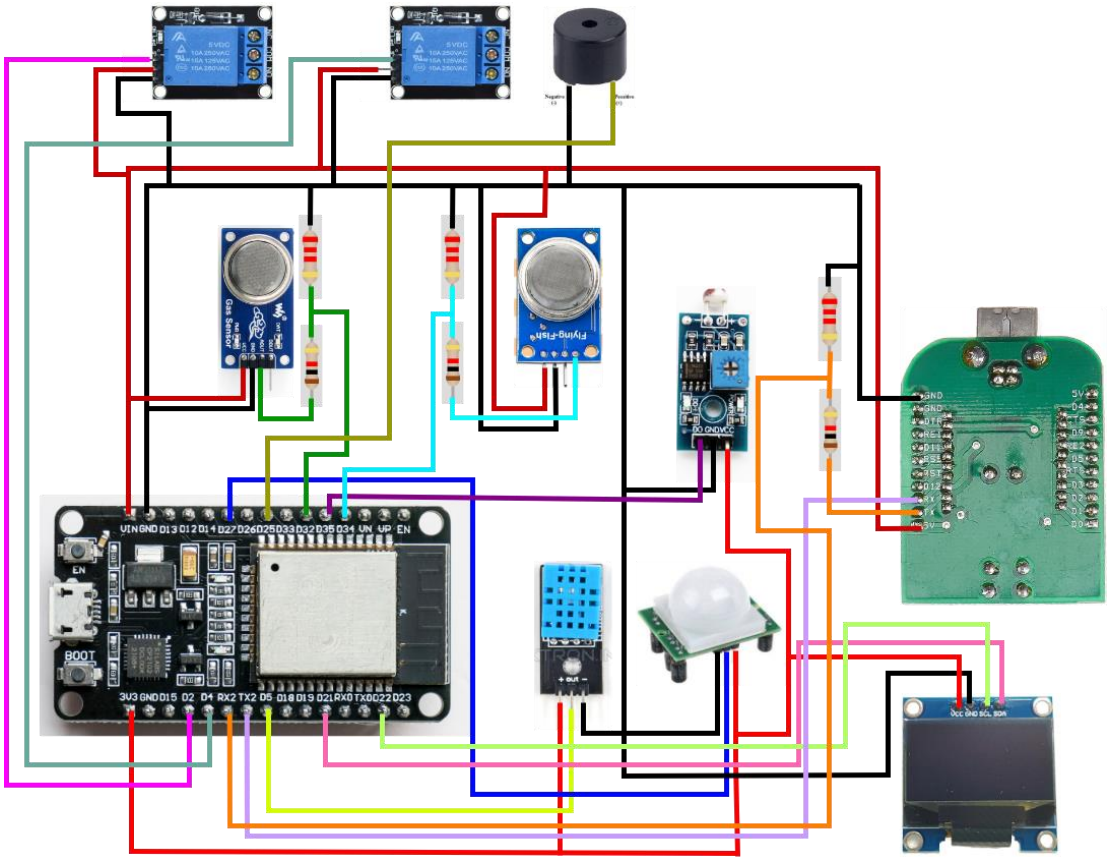

Fig. 15 Home Automation Setup

Adafruit IO Dashboard Setup

- In the Adafruit IO, we created a dashboard to visualize all the sensor data that are being sent by our system and to send all the aforementioned messages to our system.
- Firstly, we created a data section for normal visualization of all the sensor data, normal gauges are used to visualize CO2 level, room temperature & soil temperature. Linear Gauges are used to visualize soil moisture content, room humidity & light intensity. Indicators are used to visualize motion detection and smoke detection, motion detection indicator remains red for no motion detected & goes to green on detecting motion, whereas, smoke detection indicator remains green when no smoke is detected & goes to red on detecting smoke.

- Then, we created a data analytics section, where all the sensor data, except for smoke & motion sensors, are displayed in graphs for data collection purposes or finding out the conditions of home or the garden at any specific time of the day.

❖ Lastly, we created 4 switches to send command messages to the system for turning on/off lights, fans, water pumps & intruder mode.

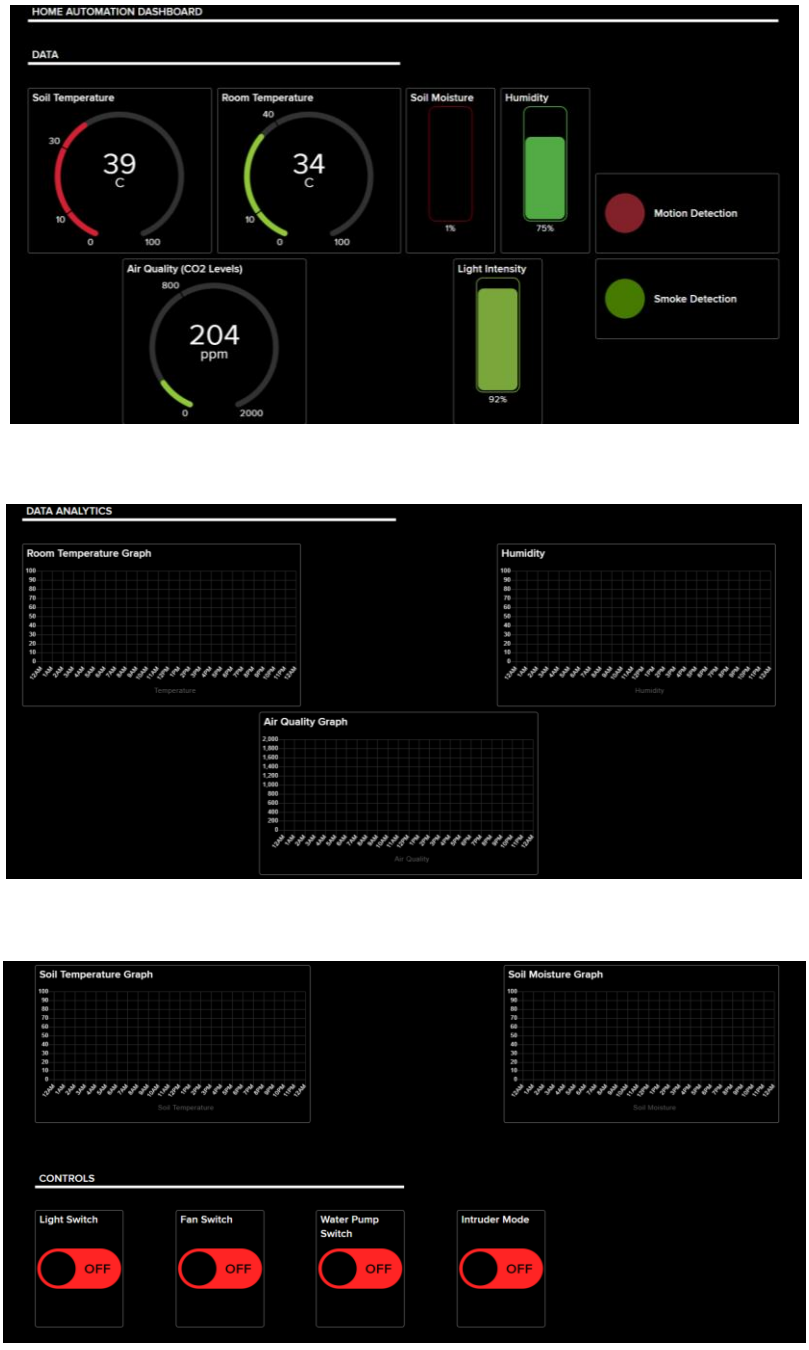

Fig. 16 Smart Home Automation & Garden Management Dashboard

Workflow

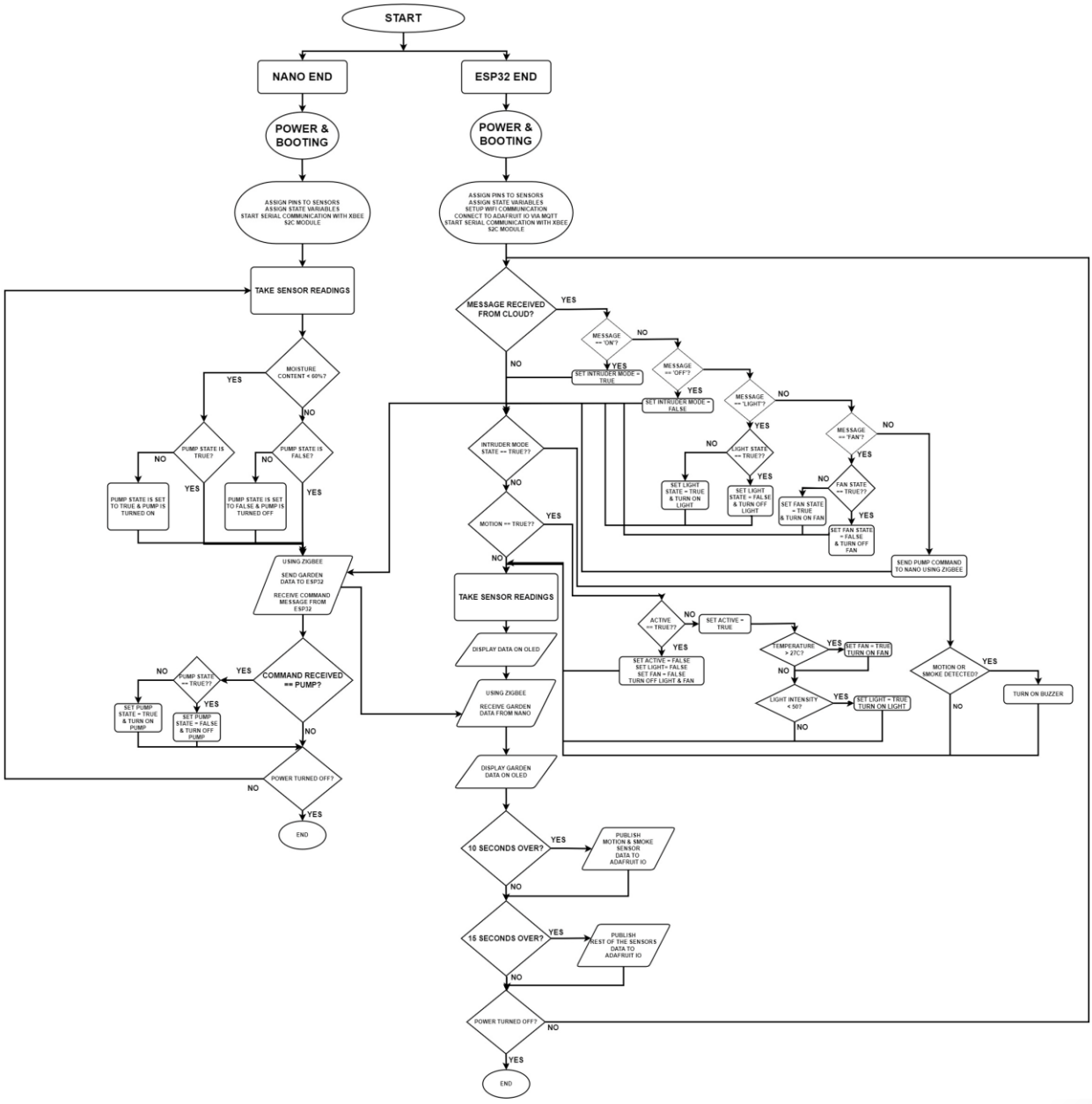

Fig. 17 Workflow of the Project

Bibliography

[1] https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-nano.html

[2] https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

[3] https://www.amazon.in/Generic-ZigBee-Adapter-Board-Interface/dp/B00PU032DM

[4] https://www.electronicwings.com/sensors-modules/xbee-module

[5] https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html

[6] https://www.componentsinfo.com/hc-sr501-module-pinout-datasheet/

[7] https://www.amazon.in/OLatus-OLSLDR005-Photosensitive-Dependent-Intensity/dp/B094RK4CXQ

[8] https://circuitdigest.com/microcontroller-projects/interfacing-mq2-gas-sensor-with-arduino

[9] https://carasalesm.best/product_details/35096877.html

[10] https://circuitdigest.com/microcontroller-projects/interfacing-soil-moisture-sensor-with-arduino-uno

[11] https://components101.com/sensors/pt100-rtd-temperature-sensor

[12] https://sumeetinstruments.com/PT100-Temperature-Sensor-Waterproof-1m-30mm-Stainless-Steel

[13] https://www.circuitschools.com/interfacing-ssd1306-oled-display-with-arduino-esp32-and-esp8266/

[14] https://www.flyrobo.in/5v_single_channel_relay_module

[15] https://learn.adafruit.com/adafruit-io/getting-started