Expertise
and insight
for the future

Trung Hoang Nguyen
Efthalia Vogiari
Thanh Tran
Cyril Mollier

Smart Plant System

Metropolia University of Applied Sciences

Information Technology - Smart Systems

IoT Project Course

Final Report

Table of Contents

List of Abbreviations

GND	Ground
GPIO	General-purpose Input/Output
I2C	Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LED	Light emitting diode
NTP	Network Time Protocol
PWM	Pulse-width modulation
RPM	Revolutions per minute
SSID	Service Set Identifier

1 Introduction

These days we are monitoring everything with our smartphone or other electronic devices. Washing machines,… or even our sleep. The Internet of Things market is booming right now.

The following report is the final document for the Internet of Things (Project) course at Metropolia University of Applied Sciences. This course challenges students to develop a useful IoTs smart product. Our team decided to focus on tree and plant lovers with a device called “Smart Pot”. For this report, the goal is to show the implementation and outcome of the product demo

2 Hardware

2.1 Components

In this project, our device can be separated into multiple components. Table 1 shows a list of components needed to ensure best performance.

Type	Component	Name	Price
Controller		Raspberry Pi 4	80e
		Arduino Uno Rev	33e
Sensor	Capacitive Moisture V1.2	EK1940	8e
	Temperature/Humidity sensor	DHT22	7e
	Light Sensor	Grove - Light sensor 1.2	3,8e

Actuators and others	Water Pumper	WayinTop Automatic Irrigation	13e

Table 1: List of final components for this project

2.2 Raspberry Pi 4

Based on the requirements of the project, we will need a small and powerful controlling unit to update every day, communicate with the server and control sensors via Wi-Fi or Ethernet. Due to its quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, the Raspberry Pi 4 could be the appropriate single-board computer for creating high-end projects requiring a board with high computational power. In addition, this device can be programmed in C, C++ and Python and can run independently as a small computer.

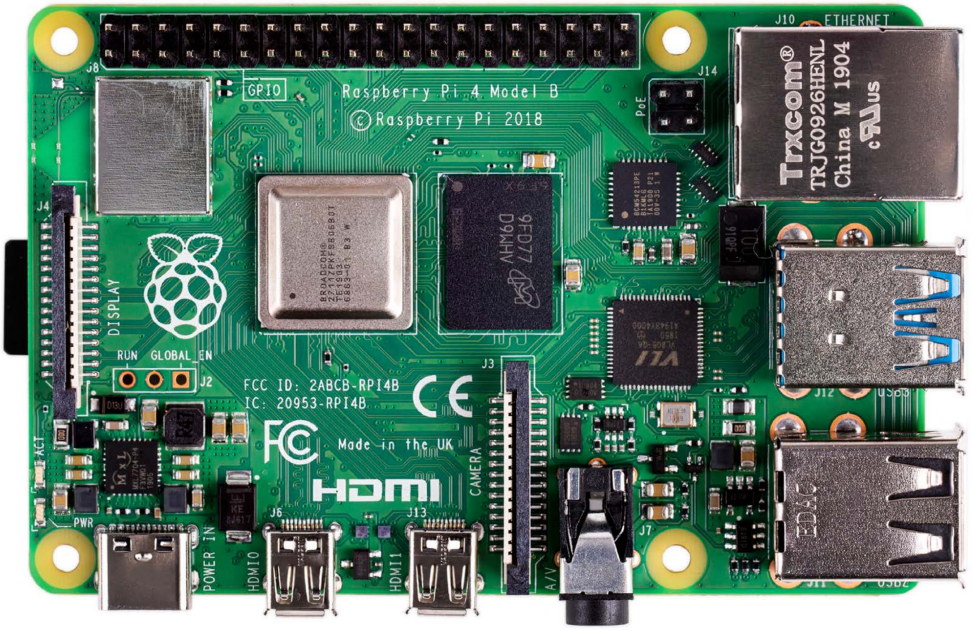

Figure 1. Raspberry Pi 4

2.3 Arduino Uno R3

Unfortunately, the Raspberry Pi has, unlike an Arduino, no built-in Analog-to-digital converter for reading analog voltage values. So, what was a very easy task on the arduino becomes a more difficult task on the Pi so in the first prototype, we chose to use Arduino as a converter. The ATmega328P-based Arduino UNO is a microcontroller board. It has 14 digital I/O pins (of which 6 can be used as PWM outputs), 6 analog inputs, a ceramic resonator operating at 16 MHz, a USB connection, a power jack, an ICSP header, and a reset button. It comes with everything you need to support the microcontroller; simply connect it to a computer via USB or power it via an AC-to-DC adapter or battery to get started. You can tinker with your UNO without fear of doing something incorrectly; worst case scenario, you can replace the chip for a few dollars and start over.

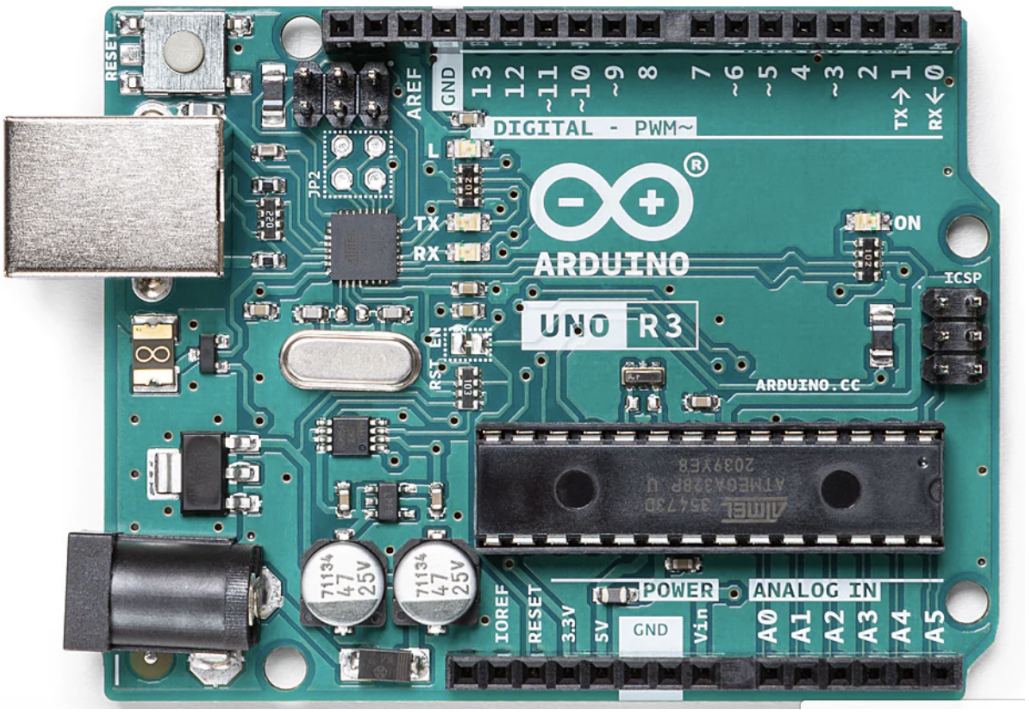

Figure 2. Arduino Rev 3

2.4 Temperature/Humidity Sensor

The DHT-22 sensor measures the relative humidity and temperature and outputs a calibrated digital output. It is a low-cost sensor that measures with greater accuracy and a wider range than the DHT11. You don't need extra components for operation. It is pre-calibrated, and you can connect it directly to get the output. The DHT22 sensor is composed of a humidity sensor and a thermistor. These two components detect humidity and temperature and transmit a digital signal to the data pin.

Figure 3. Temperature/Humidity sensor

2.5 Grove Light Sensor

To detect the intensity of light, the Grove - Light sensor incorporates a photoresistor (light-dependent resistor). When the light intensity increases, the photoresistor's resistance decreases. A dual OpAmp chip LM358 on board generates a voltage proportional to light intensity (i.e., based on resistance value). The output signal is analog; the higher the value, the brighter the light. This module can create a light-controlled switch, which will turn off lights during the day and turn on lights at night. We used this sensor to detect and measure the environment's light value.

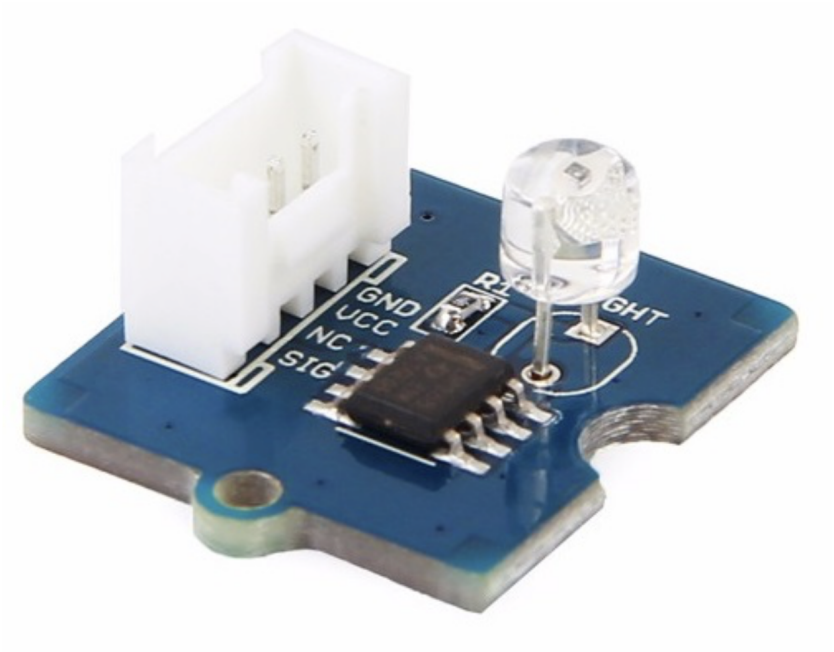

Figure 4. Light sensor

2.6 Moisture sensor V1.2

For measuring the soil moisture, we used a Capacitive Moisture Sensor. Compared to resistive sensors, capacitive sensors do not require direct exposure of metal electrodes, which can significantly reduce electrode erosion. As a result, it is referred to as Corrosion Resistant. It is important to note that

this sensor can only test the soil's humidity qualitatively and cannot measure it quantitatively. That is, as the humidity of the soil increases, the value of the output decreases; conversely, as humidity decreases, the value of the output increases..

Figure 5. Capacitive Moisture Sensor

2.7 Relay Module

A relay is an electronic switch. It has a total of five pins: Three input and two output pins.

This component is powered by Pins 1 and 2. Relays of various voltages, such as 5v, 6v, and 9v, are available. The third pin must provide the necessary potential for us to obtain the output. It can provide either AC or DC voltage. In other words, if the value is exceeded, the relay will burn out. As a result, use

devices that draw less current than the specified level. We can get outputs by using pins 4 and 5. The four-pin is known as Normally close (NC), and the five-pin is known as Normally open (NO). The four-pin connector is typically connected to the third pin. When the relay is activated, the 5th pin connects to the 3rd pin.

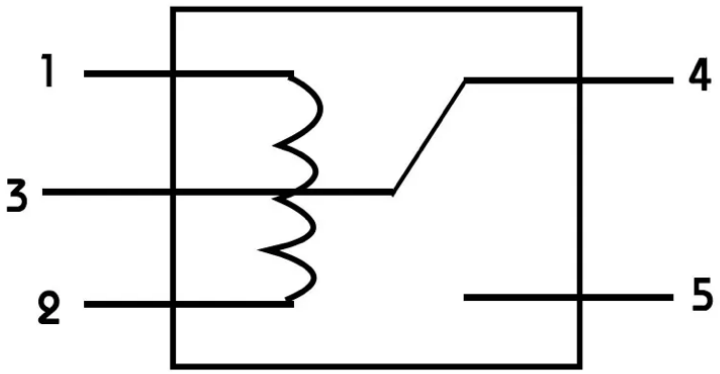

Figure 6. Relay diagram

After carefully considering, we decided to choose 5V Relay to turn on/off for our pumper.

Figure 7. Relay module for Arduino/Raspberry

2.8 Pumper

DC Mini Immersible Water Pump(figure 8) is intended to be used underwater only, which is ideal for a small watering plant. It has an inlet valve so that water inflow can be manually adjusted and a one-meter cable. The pump has a filter inside and a suction cup that helps it to stick to smooth surfaces tightly. Its power supply range is 6.0V12.0V, 65mA500mA. It pumps up to 550L/h and works quietly with a sound level under 40dB.

The problem with using the GPIO pins is that the controller only outputs a maximum of 3.3V, and that is only a high or low signal. The controller can use PWM to adjust the motor's duty cycle to the needed levels to get over this limit.

Figure 8. Pumper

3 Functional Description and Software

3.1 Block Diagram

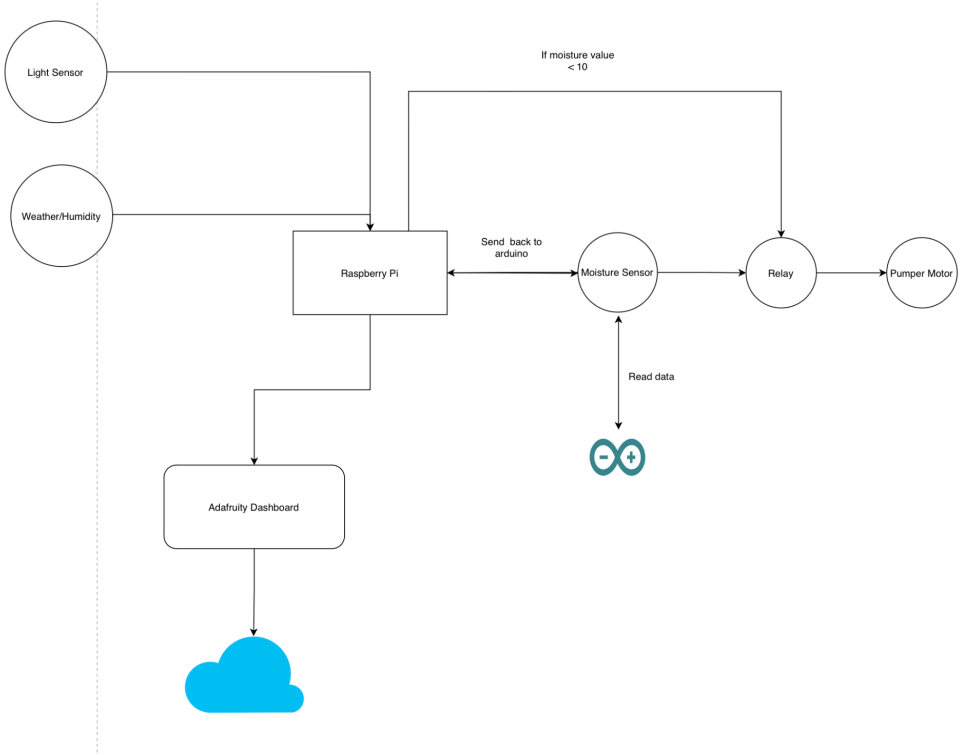

Figure 9. Project diagram

3.2 Software

There are several possible options of programming languages, platforms, and frameworks for programming on the Raspberry Pi. After figuring out our criteria and considering our possibilities, we chose Python and the Arduino IDE to control the Arduino to send analog values to Raspberry Pi via the Serial USB port.

First, Python is a straightforward language to work with. Compared to other programming languages, such as C++, in Python less code is needed in order to do the same task. Despite the ease, there are some limitations when it

comes to components. Some sensors in this project do not have existing code in Python but have higher chances to find some in C/C++. But still we managed to connect the sensors to Arduino via Micropython without any errors. Additionally, Python supports the libraries of the Adafruit IO Service that help us connect to our Adafruit IO dashboard.

The second IDE we are using, besides Geany IDE, is the Arduino IDE since there are tons of tutorials online and ready-to-go code. Its programming language is Arduino language, which is similar to C/C++, and more widely used in real life applications. Besides that it has a large user community so it is easy to fix a lot of errors without major obstacles.

3.3 Connectivity

Raspberry Pi 4 has a built-in Wi-Fi function and an Ethernet port, so depending on different usages, we can choose any kind of connection in order to optimize speed or focus on stabilization. We have already installed the user interface while installing Raspberry OS, so we can easily use it as a normal computer with a connection to the internet. In our case, we will connect via Ethernet to ensure a stable connection.

For our web platform we chose the Adafruit IO Dashboard, which is a platform for data visualization and device control. Its benefits are that it is free to use and it provides all the functionalities we need for our project.

Adafruit IO Dashboard is not too complicated to use. Once it is connected to the platform, the Adafruit IO Dashboard allows us to visualize data collected via the sensors with graphs and charts. There are built in buttons and switches to trigger particular features on our device. Furthermore, there are plenty of tutorials, well documented client libraries, and ready-to-go code available, simplifying the programming process.

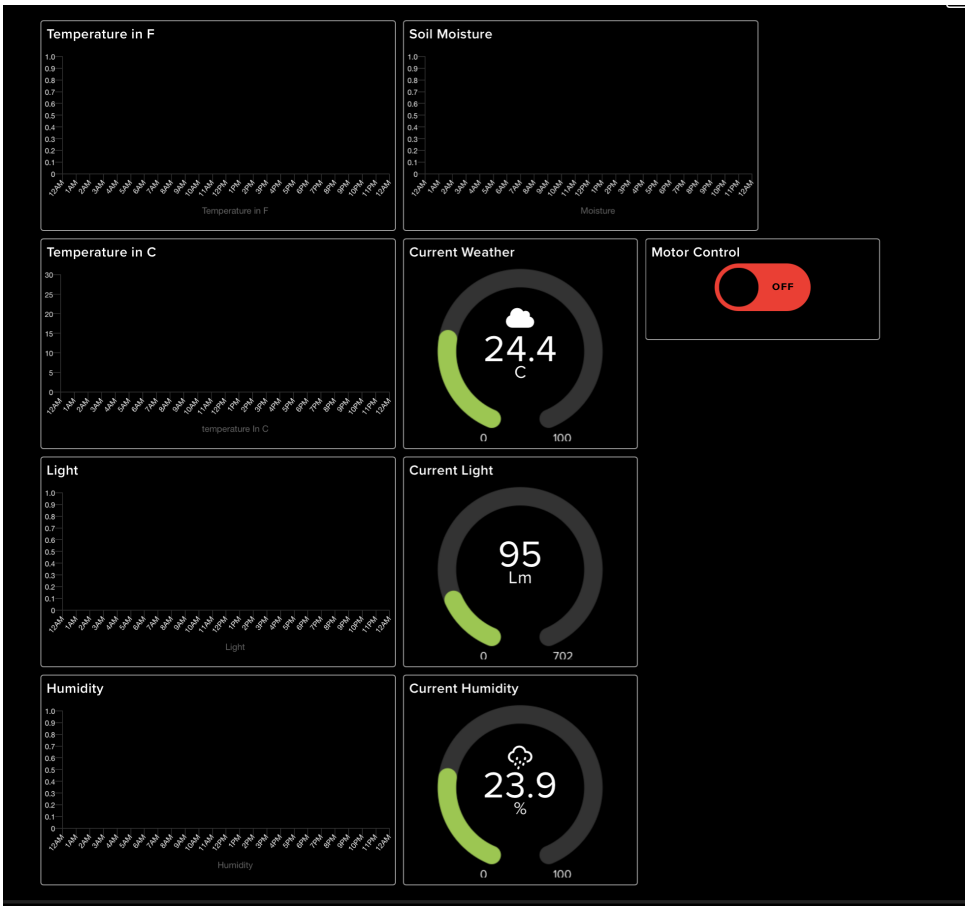

Figure 10. Adafruit Dashboard

As you can see in Figure 10 we have several blocks on our dashboard. Each block is implemented for a specific purpose such as showing current weather or humidity. Besides that it saves the values and provides the user with a graph in order to easily track the change in values or on/off switch for the water pump.

To initial the Adafruit Dashboard for the first time, we need to connect our device via the internet with the token key given by Adafruit IO.

```
# setup adafruit
ADAFRUIT_IO_USERNAME = 'cyrilmetropolia'
ADAFRUIT_IO_KEY = 'aio_FLVx78zG4clK1mH3fstd8ZOVEWO1'  # Token key for Ada Dashboard
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
```

Figure 11. Setting up dashboard

3.4 DHT22 sensor

This sensor will be attached to the Raspberry Pi via a GPIO pin. The sensor has a 3-pin connection at the side or bottom. One pin will be the ground, another will be for signal and the last pin will be for the power, which is usually up to 5V.

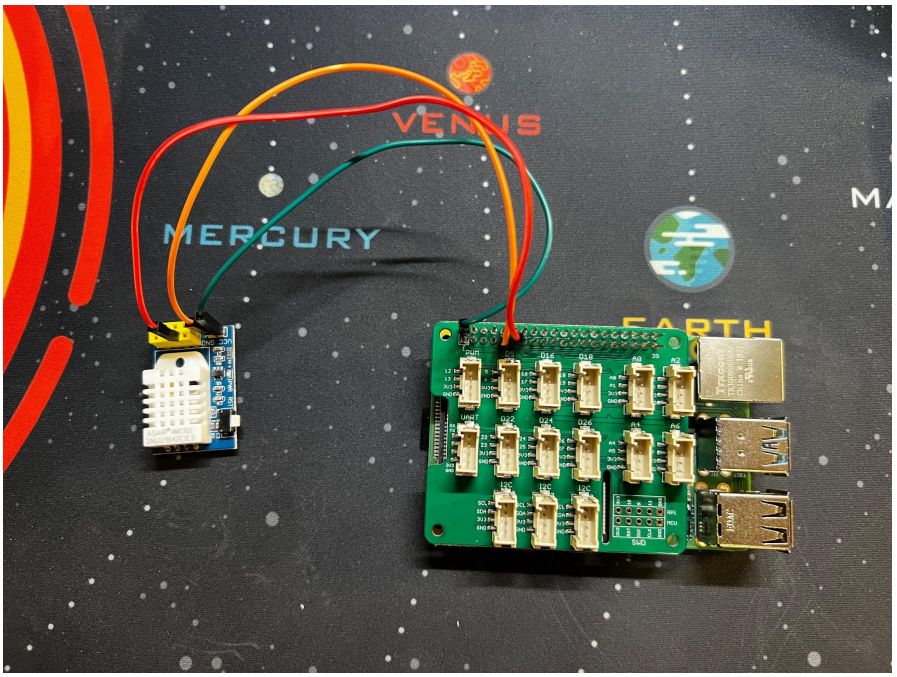

Figure 12. Connect DHT22 sensor to Raspberry Pi

The manufacturer has already prepared a library, so we only need to clone and change the GPIO with the correct ones. We used GPIO-pin 27 to send and receive values from the sensor.

```
def getTemp_Humid():
    values = []
    while True:
        try:
            dhtDevice = adafruit_dht.DHT22(board.D17,use_pulseio=False)
            temperature_c = dhtDevice.temperature
            temperature_f = temperature_c * (9 / 5) + 32
            humidity = dhtDevice.humidity
            values.append(temperature_c)
            values.append(temperature_f)
            values.append(humidity)
            print("Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(temperature_f, temperature_c, humidity))

        except RuntimeError:
            break

    return values
```

Figure 13. Function to get temperature and humidity values

3.5 Grove light sensor

Connecting the light sensor to the Raspberry Pi adapter through port A0 is needed in order for us to be able to read analog values from the light sensor.

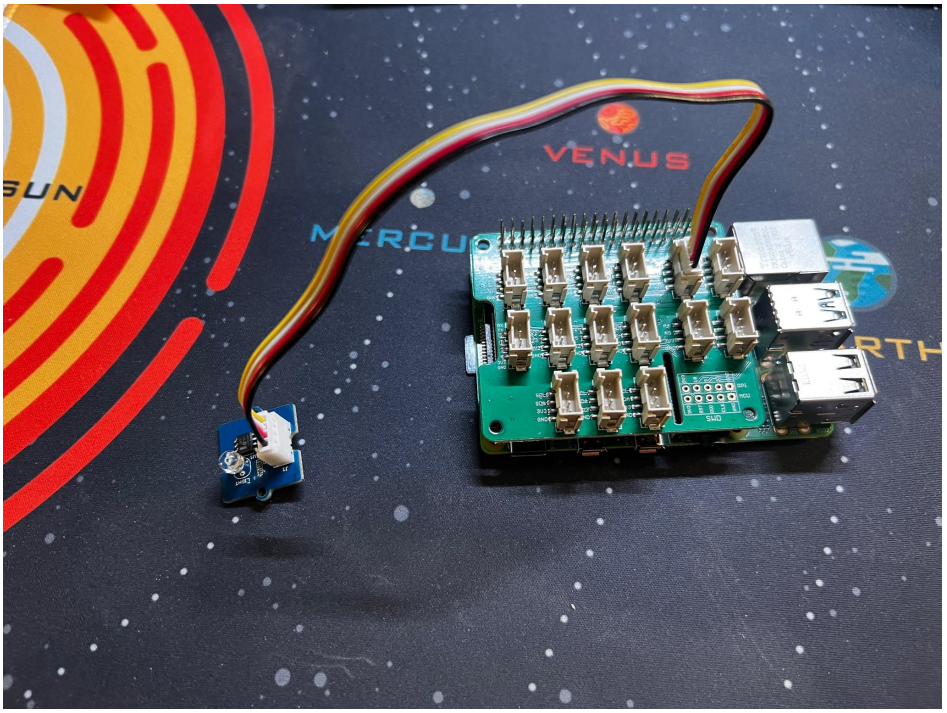

Figure 13. Connect light sensor to Raspberry

```
class GroveLightSensor:

    def __init__(self, channel):
        self.channel = channel
        self.adc = ADC()

    @property
    def light(self):
        value = self.adc.read(self.channel)
        return value


Grove = GroveLightSensor
```

Figure 14. LightSensor Class

```
def getLightSensor():
    channel = 0

    sensor = GroveLightSensor(channel)

    print('Light value: {0}'.format(sensor.light))

    return sensor.light
```

Figure 14. Function to read light sensor values

3.5 Pump

The use of the pump is to control water flow, so we will connect Raspberry Pi to the pump via Relay. Three Pins, Ground, VCC, and IN, will be connected to provide power and send/receive signal from relay to raspberry pi. The power required is 5V, so we have to plug in the same 5V GPIO pin. The motor has 2 wires, black and red. We just need to connect them to NC and COM ports on the relay. In addition, a LED indicator on the relay also indicates for the user if the relay is activated or disabled.

```
def PumperOn():
    GPIO.setup(Relay_GPIO, GPIO.OUT)  # Initial GPIO 21
    GPIO.output(Relay_GPIO, GPIO.HIGH)  # Turn Relay On
    print("Plant is watering")
    time.sleep(5)
    GPIO.output(Relay_GPIO, GPIO.LOW)  # Turn Relay Off
    print("Watering is finished")
```

Figure 16. Water pump function

3.6 Moisture sensor

This is one of the most challenging parts in this project because the Raspberry Pi can not read analog values. We decided to use an Arduino to connect to the moisture sensor and send back the values to the Raspberry Pi via a USB serial port.

As mentioned, the data can be transferred using serial communication. In contrast to parallel communication, which sends many bits at once, the data will be transmitted sequentially, one bit at a time. It basically works as an asynchronous multi-master protocol that will let you talk to both boards at once. All connected devices will be able to send data whenever they want, thanks to it being a multi-master protocol. This is a key distinction from master-slave protocols, in which only the master device can start a conversation.

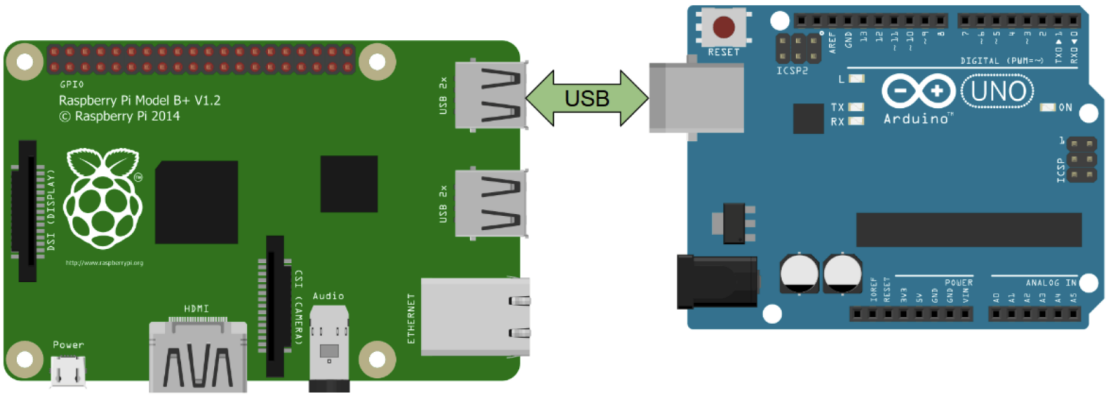

Figure 17. Connection between Raspberry Pi and Arduino

```
def getMoistureSensor():
    ser = serial.Serial('/dev/ttyUSB0', 9600)
    ser.flush()  # Clear cache
    moisture_value = 0
    while True:
        try:
            if ser.in_waiting > 0:
                line = ser.readline().decode('utf-8').rstrip()
                print(line)
                moisture_value = re.findall("-?\d+", line)
                if len(moisture_value) == 1:
                    moisture_value = moisture_value[0]
                    moisture_value = int(moisture_value)
                    if moisture_value < 0:
                        print("Plant need water")
                        PumperOn()
                    else:
                        print("Plant is full")
                    break

        except serial.serialutil.SerialException:
            pass

    return moisture_value
```

Figure 17. Function to read values from Moisture Sensor

To process the data of the moisture sensor following steps are made: First the Arduino gets the data. In order to receive and process data, a while-loop has been created. The third stage of the script will then make a loop condition to cause the pumper to run if the value is below 10, after which it will filter out the correct value and convert it to a string.

3.7 Assembly

The Raspberry Pi's memory stores all of its previously used functions in individual files. We need to isolate each component so that we can make changes or correct errors without disrupting the operation of the others. The IDE and PC are not required for this automatic operation because a while-loop has been set up. Getting power to the Raspberry only requires a USB-C cable.

```python
if __name__ == '__main__':

    while True:

        # send temp/humidity values to adafruit
        tempHumid = getTemp_Humid()
        if not tempHumid:
            print("list is empty")
        else:
            aio.send_data(tempC.key, tempHumid[0])
            aio.send_data(tempF.key, tempHumid[1])
            aio.send_data(humidity.key, tempHumid[2])

        # send light values to adafruit
        aio.send_data(light.key, getLightSensor())

        # send moisture data to adafruit
        moistureData = getMoistureSensor()
        time.sleep(3)
        if not moistureData or moistureData == 0:
            print("list is empty")
        else:
            aio.send_data(moisture.key, getMoistureSensor())

        print(moistureData)

        time.sleep(4)  # so we dont exceed the adafruit request limit so fast
```

Figure 18. Main method

The script will run automatically every time the device is turned on and values will be sent to the dashboard. As can be seen from the main scripts, we made functions for each component, then we will recall them and send the value to server. “Time.sleep” command is used to make sure that sensors can cool down after xxx seconds so that we can receive the most accurate values before sending to the server

4 Problems

Issues	Solutions
Remote working due to different schedule	Divide the works as much as possible. But in reality, the person holding the controller was the one to be able to properly develop at a time.
Raspberry Pi only reads digital signals	We needed to use an adapter or an Arduino to convert from digital to analog. We chose to use the Arduino.
Difficulty when building an application and visualizing values. Free version can not send/receive data every second.	Use a ready-made platform, We send data every 4 seconds, in order to not exceed the request-limit.
DHT22 temperature/humidity sensor sometimes does not send any values	Use a try-except block, to manage Runtime Exceptions.

The original plan was to have 6 features: monitor temperature, humidity, monitor environment light to adjust a LED lamp, monitor moisture, control watering time based on moisture values. The team managed to finish 5 out of 6 features (except for adjusting LED level). In conclusion, the team agreed that the Smart Pot project achieved the initial objective and were happy with the results.

Still, some improvements can be carried out in this project. Some additional features discussed in the beginning, such as the ability to adjust the light level or to control the watering time itself, were unfortunately not considered in this stage due to their extreme complexity. Other upgrades can be done in the future so that the final product can be commercially or widely used.

5 Conclusion

It was an invaluable experience for the team to do this project, as we got excellent opportunities to put what we have been learning so far about IoT and embedded systems into practice: coding with Raspberry Pi and Arduino, handling sensors, implementing M2M communications, and putting them together to build a full IoT device. We are thankful to receive enthusiastic support and constructive comments from the teacher. The project can be carried on in the Innovation Project course.

Sources

Raspberry Pi 4. Raspberry official store. Available from: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/> (cited November 25 2022)

DHT22 Temperature/Humidity Sensor. Adafruit official store. Available from: <https://www.adafruit.com/product/385> (cited November 20 .2022)

Grove Light Sensor. Digi-key electronics. Available from: <https://wiki.seeedstudio.com/Grove-Light_Sensor/> (cited November 19 .2022)

Capacitive Moisture Sensor. Amazon Shop. Available from: <https://www.amazon.com/HiLetgo-Capacitive-Moisture-Voltage-3-3-5-5V/dp/B07W83ZVFB?keywords=capacitive+soil+moisture+sensor+v2.0&qid=1670424686&sprefix=capacitive+soi,aps,96&sr=8-2&linkCode=sl1&tag=howtoelect0e4-20&linkId=becd1babdab3644cd9e500a9c4ef6159&language=en_US&ref_=as_li_ss_tl> (cited November 16 2022)