



Expertise  
and insight  
for the future

Linda Akosua Dankwa

# Cost effective EC and pH measurements

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics Engineering

Bachelor's Thesis

3 June 2020

Author Title Number of Pages Date	Linda Akosua Dankwa Cost effective EC and pH measurements 36 pages + 4 appendices 3 June 2020
Degree	Bachelor of Engineering
Degree Program	Electronics Engineering
Professional Major	Electronics
Instructors	Antti Liljaniemi, Senior Lecturer Janne Mäntykoski, Senior Lecturer
<p>Often times, it is difficult to find electrical conductivity and potential of hydrogen measurement sensors with PLC compatible interfaces at a low cost. The reason for this might be due to the fact that the companies producing industrial grade sensors for continuous measurement are extremely scarce and can therefore dictate the market value. Another reason may simply be because it is more expensive for the device manufacturers to make items to order instead of producing them in bulk.</p> <p>This thesis was conducted for Metropolia Urban Farm Lab, a separate co-learning space for experts, students and companies alike to find solutions for improving the well-being of humans on earth. The main goal of this thesis was to explore cost effective methods for measuring, recording and monitoring the electrical conductivity and potential of hydrogen of the watering solution for hop plants.</p> <p>After conducting research through multiple avenues (both online and in the real world), it was deduced that not only were cost effective EC and pH measurements possible, but could also cost approximately half the price that a majority of companies are currently paying. By following the datasheets and specifications of the different devices (provided by their manufacturers), simulation models that closely mimic how the devices would act in real life were created and the data they produced was analysed.</p> <p>MindSphere (an open, cloud-based IoT operating system created by Siemens) was used to record and graph the values detected by the cost effective measurement system. The IoT platform made it significantly easier to monitor the system from afar, since the platform can be accessed (by any authorised personnel) from anywhere in the world.</p> <p>Future areas of research and the impact of the findings of this project are also given in the conclusion of this thesis.</p>	
Keywords	Electrical conductivity, EC , pH, Raspberry Pi, cost effective, MindSphere, MindConnect, IoT

## Contents

### List of Abbreviations

1	Introduction	1
2	Electrical Conductivity	2
3	Potential of Hydrogen	6
4	Data measurement	9
4.1	Measurement circuit	9
4.1.1	Circuit diagram	15
4.1.2	Raspberry Pi 3 configuration	17
4.1.3	Sensor calibration	20
4.1.4	Circuit Simulation	23
4.1.5	Results	25
4.2	Mindsphere	28
5	Conclusion	33
	References	34

### Appendices

Appendix 1. The main Raspberry Pi code

Appendix 2. EC and pH meter Raspberry Pi codes

Appendix 3. The EC and pH calibration files

Appendix 4. How to connect a device to MindSphere(MindConnect)

## List of Abbreviations

EC	Electrical Conductivity. The ability of a substance to transmit electricity.
PH	Potential of Hydrogen. The measure of the acidity or alkalinity of a solution.
PLC	Programmable logic controller. An industrial computer control system that continuously monitors the state of input devices and makes decisions based upon a custom program to control the state of output devices.
SI	Système International. The international system of units used for scientific measurements.
K	Cell Constant. It is directly proportional to the distance separating the two conductive plates and inversely proportional to their surface area.
PTFE	Polytetrafluoroethylene. A synthetic fluoropolymer of tetrafluoroethylene.
ADC	Analogue to Digital Converter. A system that converts and analogue signal to a digital signal.
VCC	Voltage Common Collector. The voltage at the collector of a device and the power input of a device.
VDD	Voltage Drain Drain. The voltage at the drain of a device.
GND	Ground. The reference point in an electrical circuit from which voltages are measured, a common return path for electric current, or a direct physical connection to the Earth.
Vout	Voltage output. The output voltage of a device.
GPIO	General-Purpose Input/Output. An uncommitted digital signal pin on an integrated circuit whose behaviour is controllable by the user at run time.
VREF	Voltage Reference. A precision device designed to maintain an accurate, low noise, constant output voltage.

AGND	Analogue Ground. The path for analogue circuitry to return to the ground.
DGND	Digital Ground. The path for digital circuitry to return to the ground.
DOUT	Digital Output. A signal used to control items with only two states, on high and low.
DIN	Digital Input. A signal that allows devices to detect logic states.
CLK	Clock. A clock signal fluctuates between a high and a low state and is used as a metronome to coordinate the actions of digital circuits.
CS	Chip Select. A control line in digital electronics used to select one (or a set) of integrated circuits out of numerous connected to the same computer bus, usually utilizing the three-state logic.
MISO	Master In Slave Out. The Slave line for sending data to the master.
MOSI	Master Out Slave In. The Master line for sending data to the peripherals.
URL	Uniform Resource Locator. An address for a particular page on the World Wide Web.
MQTT	Message Queue Telemetry Transport. A messaging transport protocol, that uses the brokered publish/subscribe system.
BNC	Bayonet Neill–Concelman. A miniature quick connect/disconnect radio frequency connector used for coaxial cables.
SPI	Serial Peripheral Interface. A synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances
API	Application Programming Interface. A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

## 1 Introduction

Hops plants have a large variety of uses and benefits. They have medicinal compounds that can act as a mild sedative for the treatment of insomnia, they can be used to create “Hops pillows” which are helpful in inducing sleep, and are occasionally used in culinary dishes due to their unique and incomparable taste. An outdoor grown hop is also a key ingredient in the production of an undeniably aromatic, earthy and flavourful beer. Unfortunately, this type of beer can only be brewed during hop harvest season: an unbelievably short season that occurs once a year. In a climate controlled cultivation process, hydroponic hop crops can be maneuvered in a way that they would be able to supply high-quality fresh crops throughout the year, after their successional planting. There have been numerous reports that with the use of controlled nutrition in hydroponics, not only can hops with higher concentrations of essential oils, aromatic compounds, beta acids, and flavonoids be produced, but also ones with larger and heavier cones (hop plant flowers) with a significantly higher overall yield [1].

HydroHumala is an indoor hydroponic hops plant farming solution for year round production of fresh hops for breweries [2]. It is done in collaboration with Tornion Panimo’s brewery and Metropolia University of Applied Sciences (Urban farmlab) and is estimated to produce about 3000 litres of a new type of beer using 100% fresh hops. With such a significant investment on the line, it is extremely imperative that the conditions for the optimum growth of the plants are maintained. Keeping the electrical conductivity (EC) and the pH of the liquids used to water the plants at the perfect level for plant growth is a top priority. For this to transpire, continuous EC and pH measurements need to occur with the use of industrial grade equipment. This will not only guarantee that reliable measurement values are obtained, but also ensure that the sensors will not have to be replaced too frequently (simply due to water damage).

Reliable industrial grade sensing circuits and equipment can all too often cost thousands of euros at a time. The aim of this thesis is to provide a cost effective method for measuring, recording and analysing data from sensors in addition to creating an IoT platform for easy and convenient data visualisation.

## 2 Electrical Conductivity

The electrical conductivity (EC) of a substance is the ability of a substance to transmit electricity. Its units are Siemens per meter [S/m] in SI and millimhos per centimeter [mmho/cm] in U.S. customary units. An electric current arises from the movement of electrically charged particles in response to forces that act on them from an electric field. For most solid materials, the current is produced by the movement of electrons and is called electronic conduction. In all conductors, semiconductors, and many insulated materials only electronic conduction exists, and the electrical conductivity is strongly dependant on the number of electrons available to participate to the conduction process. Due to the high number of free electrons that can be stimulated in an empty and ready energy state, most metals are exceptionally good conductors of electricity. In fluids and ionic materials, a net motion of charged ions called “ionic conduction” can occur. Electrical conductivity is defined as the ratio between the current density (J) and the electric field intensity (e) and it is the opposite of the resistivity (r) .[3.] The equation for electrical conductivity (1) is seen below.

$$E. C = \frac{J}{e} = \frac{1}{r} \quad (1)$$

In fluids, electrical current is transported by the available ions. This means that the conductivity of a fluid would increase as the concentration of ions in the substance increases. The data shown in Table 1 exemplifies this phenomenon. Ultra-pure water (which has no nutrients or minerals) has an EC value of about zero and would therefore be a very poor conductor of electricity. Sea water on the other hand (a liquid containing a large quantity of minerals) has an EC value of 5 and would therefore be a good electrical conductor.

Table 1. The typical conductivity of different solutions [3].

Substance	Electrical Conductivity (S/m)
Ultra-pure water	$5.5 \cdot 10^{-6}$
Drinking water	0.005 – 0.05
Sea water	5

In the growth and production of Hops plants, electrical conductivity measurements can help determine the strength of hydroponic nutrient solutions. Formulas that are high in salts can impair the plants ability to absorb water, which would have a negative impact on plant growth. Unusually high EC values can also be an indication of cross contamination from salt dense fertilisers and should always be monitored closely.

Table 2. The salt tolerance of plants belonging to different water groups [4].

<b>Water group</b>	<b>Salt sensitivity</b>	<b>Optimum conductivity range (mS/m)</b>
A	Highly salt sensitive	0–90
B	Mildly salt sensitive	90–270
C	Slightly salt sensitive	270–635
D	Salt tolerant	635–2365

The sensitivity of plants to salt can differ substantially. In Table 2, the plants are organized in an estimated order of salt tolerance for each category, the least tolerant being described first. These plant and water groups are only a general guide, as soil texture and drainage could be overriding factors. [4.]

Most plants can tolerate saline solutions between 150 – 350 mS/m. Hops plants however, have a salt sensitivity of “fair” [5]. These types of plants belong to water group B (making them only mildly salt sensitive) and thrive especially well when the solution used in watering the shrubs has a conductivity of between 90–270 mS/m [4]. Exposing the Hops to conditions outside this range may be catastrophic, so the best range of EC values to maintain to ensure that there will be a bountiful harvest would be ones that are far enough away from the hard limits (to allow room for sensor related measurement errors). In this case, keeping the saline solution between 150 - 250mS/m would be the best course of action.

To keep the EC at the appropriate range, industrial grade sensing circuits with the ability to perform continuous measurements need to be utilized. This is done by connecting sensing electrodes (or a probe) designed for conductivity measurements to an EC meter.



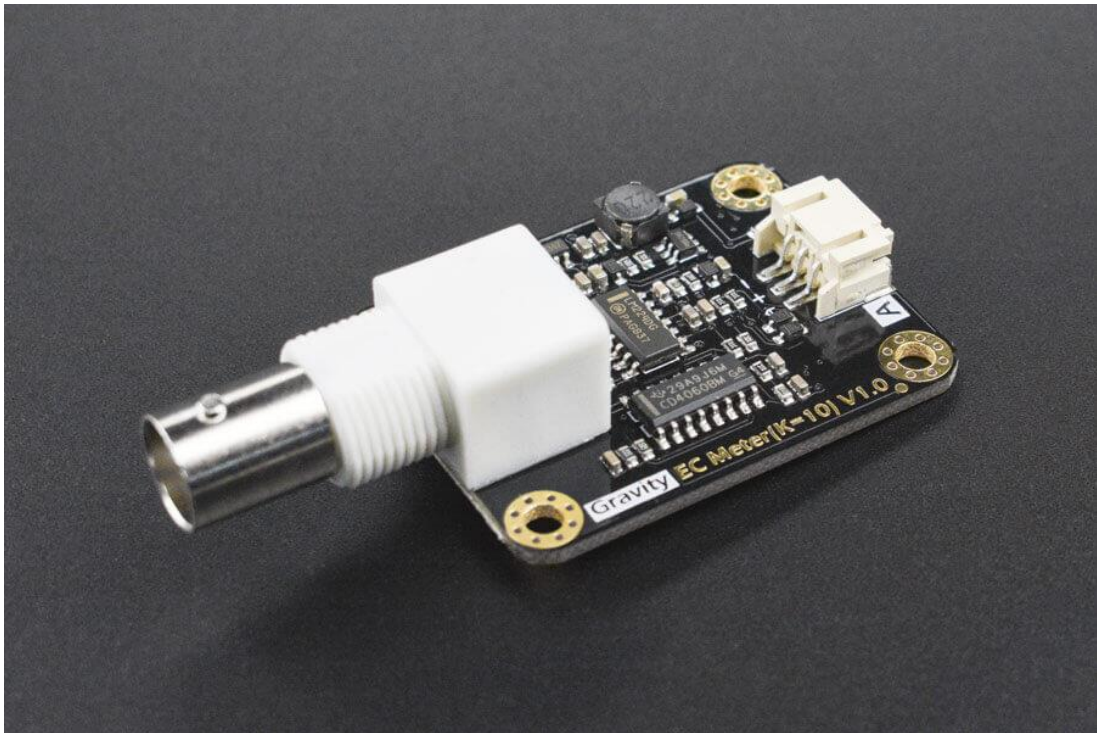


Figure 1. A DFRobot analogue EC meter [12].

DFRobot, a world-leading robotics and open source hardware provider has a particularly good electrical conductivity meter in stock (shown in Figure 1). It is not only designed to provide a great user experience, but also unequivocal measurement precision [12]. It can be used to measure high conductivity values, supports a 3~5v wide voltage input and is compatible with 5V and 3.3V main control boards like Raspberry Pi's [9]. The output signal is filtered by the hardware and has low jitter. The excitation source adopts an AC signal, which effectively reduces the polarization effect, improves the precision and prolongs the life of the electrodes. [12.]

Table 3. Alvin Instrument EC electrode specifications [11].

Cell constant (K)	Sensing Material	Conductivity Range (mS/m)
0.01	Stainless steel	0.001~2
0.1	Stainless steel	0.01~20
1.0	Platinum	0.1~200
10	Platinum	1~2000

Unfortunately, DFRobot does not produce industrial grade probes or electrodes for continuous EC measurement; these devices need to come from another source. At only € 40.93 apiece, Alvin Instrument has established itself as a company that produces both affordable and good quality sensing electrodes [11]. Their electrodes (shown in Figure 2) are suitable for measuring a wide variety of fluids and liquids. The conductivity range of the electrodes chosen depend heavily on the cell constant: the higher the K value, the wider the conductivity range of the electrodes.



Figure 2. Industrial grade two-pole type electrodes for continuous conductivity measurements with a platinum sensing material and a cell constant of 10 [11].

As previously mentioned, the EC of the watering solution needs to be around 150 and 250mS/m at all possible times. By studying the information available on Table 3, it is abundantly clear that the only electrodes that are capable of measuring values in this range are the electrodes with a cell constant of 10. For this reason alone, the industrial grade two-pole type electrodes for continuous conductivity measurements (with a platinum sensing material and a cell constant of 10) will be used for this project.



it would be wise to keep both the soil and the watering solution at a slightly acidic pH (between 6.7 and 6.9). Values in this range are a good distance away from the cut-off points and will make room for sensor related measurement errors.

To keep the pH at the appropriate range, industrial grade sensing circuits with the ability to perform continuous measurements need to be utilized. This is done by connecting sensing electrodes (or a probe) designed for pH measurements to a pH meter.



Figure 4. A DFRobot analogue pH meter [10].

Unlike what was done when selecting the appropriate devices for accurate EC measurements, the pH meter and the industrial grade electrode can both be provided by one supplier. DFRobot has a pH sensor meter kit that comes with an industrial grade pH electrode (Figure 5), a pH meter (Figure 4), an analogue cable and a BNC connector [10].



Figure 5. A DFRobot pH electrode [10].

The pH electrode used in this project is seen in the figure above. It is built using a delicate glass membrane with a relatively low impedance. It is often utilized in a wide range of pH measurements due to its expeditious response and excellent thermal stability. It is extremely reliable, does not break down after being continuously submerged in water for a prolonged period of time and can eliminate a plethora of basic alkali errors. It also has a linear output voltage when subjected to solutions in the 0pH to 14pH range. Its reference system consists of an Ag/AgCl gel electrolyte salt bridge, which is known to have exceptional anti-pollution performance and have a relatively stable half-cell potential. Additionally, it has a PTFE ring that cannot be obstructed easily and is therefore suitable for long-term online detection.[10] With this many useful features, this electrode is fairly capable of measuring the pH of watering solutions and ensuring that they remain between 6.7 and 6.9.

## 4 Data measurement

### 4.1 Measurement circuit

After identifying the sensors needed and the range of EC and pH values they are required to be able to measure, the next logical step is to create a basic layout of the overall circuit and connections.

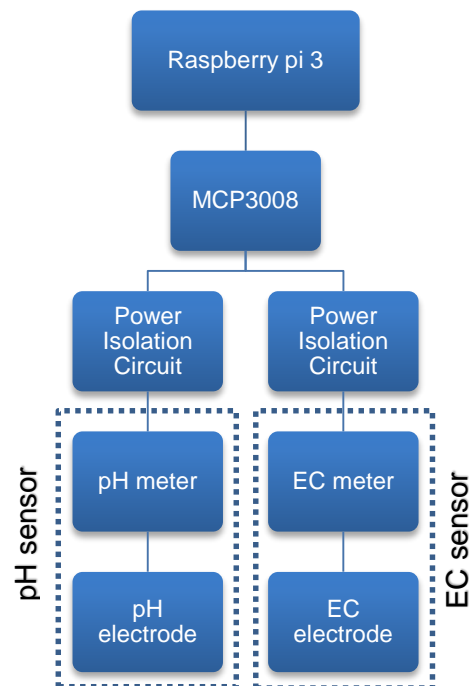


Figure 6. A basic layout of the circuit diagram and the component hierarchy.

As seen in Figure 6, the circuit contains five major parts

- A Raspberry Pi 3 (A microcontroller)
- A MCP3008 (An ADC)
- Two power isolation circuits
- A pH sensor (comprising of a pH electrode and meter)
- An EC sensor (comprising of an EC electrode and meter)



The EC and pH sensors are the first point of contact between the circuit and the outside world. After coming into contact with different solutions, they each produce a very specific voltage value that corresponds to the pH or EC value of the solution. The data is then sent to the meters, which amplifies and scales the values, dramatically increasing the differences between voltage values which were originally close together.

Electrical power isolation is required in circuits containing sensors in order to guarantee that the circuit (as a whole) will function as intended. EC and pH sensors have been known to interfere with each other and measure incorrectly when they are connected to the same power source. To eliminate interference between the two sensors, it is necessary to isolate the power sources and the signal produced by the sensors. A cost effective method of achieving power isolation is by making use of a device called an opto-isolator. An opto-isolator is a device used to transfer electrical signals between two isolated circuits by using light. Conventional opto-isolator circuits are designed on photo diode and phototransistor based networks. These circuits however, are not suitable for the direct isolation of analogue signals and are generally used for isolating digital signals. In order to isolate an analogue signal, a circuit resembling the one seen in Figure 7 is required. [23, 6.]

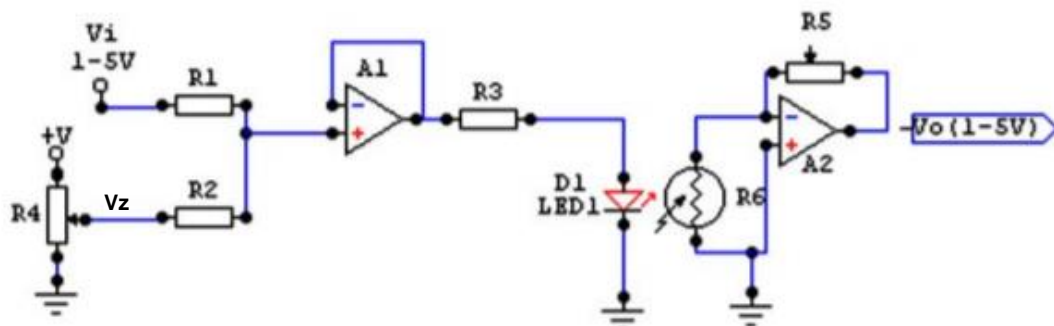


Figure 7. A low cost opto-isolator circuit for isolation of a d.c. analogue signal [23, 6].

In Figure 7, R6 signifies a light dependent resistor (LDR), D1 denotes a light emitting diode (LED), Vi signifies the input analogue signal, Vz signifies a zero adjustment signal, A1 signifies a summing operational amplifier at the input and A2 signifies a gain adjustment operational amplifier at the output. [23, 7.]

When an analogue d.c. current is passed through a light emitting diode in the forward bias condition, the intensity of light emitted from the LED is linearly correlated with the d.c. current passing through it. When this light shines on a light dependent resistor that is linearly correlated to light intensity, the resistance of the LDR decreases with the increase in light intensity. Hence the resistance of the LDR unit will decrease with an increase in the d.c. current passing through LED. So for the series resistance  $R_3$ , the d.c. current passing through the LED is given by the following equation (2): [23, 7.]

$$I = \frac{V_i + V_z - V_{LED}}{R_3} \quad (2)$$

Now the light power  $P_L$  emitted by LED is directly proportional to this current. Hence, [23, 7]

$$P_L \propto I$$

or

$$P_L = \frac{K_1(V_i + V_z - V_{LED})}{R_3} \quad (3)$$

Where  $K_1$  is a proportionality constant. [23, 7]

One end of the LDR is connected to the -V volts potential terminal of a stabilized d.c. source and the other to the virtual ground. Hence if  $I_d$  is the current through the LDR in the dark condition and  $I_{hp}$  is the current in the lighted condition, then the resistance of the LDR is given by the following equation (4): [23, 7.]

$$R_{LDR} = \frac{V}{(I_{hp} + I_d)} \quad (4)$$

In a dark region,  $I_{hp}$  is zero and  $I_d$  is very small; thus, the value of  $R_{LDR}$  is very high in dark conditions. In a lit region, the incident power ( $P_{inc}$ ) on the LDR's surface is proportional to the power produced by the LDR ( $P_L$ ). [23, 7.] Hence,

$$P_{inc} \propto P_L$$



or

$$P_{inc} = K_2 P_L \quad (5)$$

where  $K_2$  is the constant of proportionality. [23, 7]

Current  $I_{hp}$  produced by incident photons in the LDR material is given by [23, 7]

$$I_{hp} = \frac{\eta q P_{inc}}{h\nu} \quad (6)$$

where  $q$  is the electronic charge,  $\eta$  is the quantum efficiency,  $h$  is planck's constant and  $\nu$  is the frequency of the light emitted by the LDR. [23, 7]

Combining equations (3), (4), (5) and (6) we get, [23, 7]

$$R_{LDR} = \frac{V}{\frac{I_d + \eta q K K_1 (V_i + V_z - V_{LED})}{R_3 h \nu}} \quad (7)$$

Hence the output voltage ( $V_O$ ) of the LDR circuit is given by [23, 8]

$$V_O = \frac{R_5 V}{R_{LDR}} \quad (8)$$

or

$$V_O = R_5 \left( \frac{I_d + \eta q K K_1 (V_i + V_z - V_{LED})}{R_3 h \nu} \right) \quad (9)$$

Thus the isolated output voltage is directly proportional to the input voltage (since all the other parameters in the above equation (9) are almost constants). [23, 8]

Hence  $V_O = V_i$ , when the following equation (10) is satisfied: [23, 8]

$$V_i \left( \frac{1}{R_5} - \frac{\eta q K_1 K_2}{R_3 h \nu} \right) = \frac{\eta q K_1 K_2 (V_z - V_{LED})}{R_3 h \nu} + I_d \quad (10)$$

This condition can be achieved by adjusting the zero adjustment voltage  $V_z$  and the gain adjustment feedback resistance  $R_5$  by trial and error. Once this occurs, an isolated output voltage (that mirrors the input voltage) can be produced by the circuit. [23, 8.]

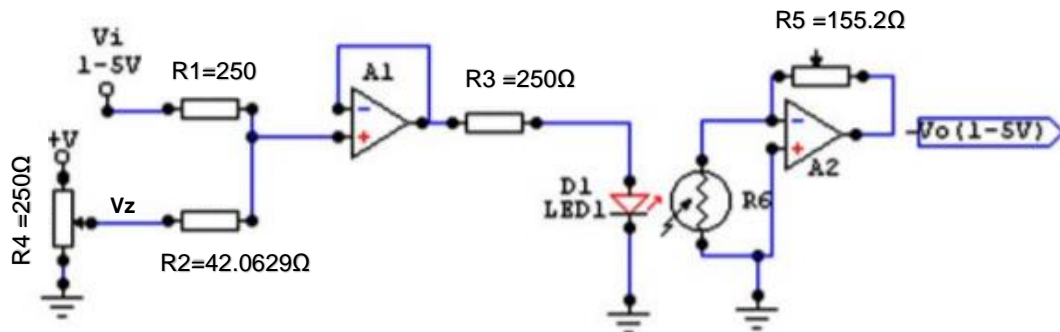


Figure 8. The opto-isolator circuit after zero and gain adjustments [23, 6].

Figure 8 shows the outcome of the zero and gain adjustments by trial and error. In order to make the input voltage directly proportional to the output,  $R_2$  had to be set to approximately 42Ω,  $R_5$  had to be set to about 155Ω and all other resistors had to be set to 250Ω.

The output of both sensors (and consequently the output of the power isolation circuits) is analogue. This would be completely irrelevant if the measurement system used an Arduino as its microcontroller (since it has several analogue pins). Instead, this system makes use of a Raspberry Pi as its main controller and consequently has only GPIO pins (digital pin in and out). Thus to be able to make sense of the data from the sensors, an analogue to digital converter needs to act like a bridge between the sensors and the Raspberry Pi. For this specific undertaking, the MCP3008 was chosen. Its precision is similar to that of an Arduino Uno and can read analogue signals from up to 8 channels [14]. It is also extremely cheap and readily available (due to its various applications). This chip is a great option for projects like this, where simple analogue signals need to be read.

Lastly, there is the Raspberry Pi. The microcontroller itself is quite atypical i.e. its cheapest version does not have a case and is a card sized electronic board (a bit smaller than those found in a PC or laptop). The quad-core Raspberry Pi 3 is quicker and more proficient at its tasks than its immediate predecessor, the Raspberry Pi 2. The device's CPU (the main processor) has an approximately 50 to 60 percent better performance in 32-bit mode than the Pi 2 and is ten times quicker than the original single-core Raspberry Pi (based on a multi-threaded CPU benchmark in SysBench). Compared to the original Pi, real-world applications see a performance increase of about 2.5x (for single-threaded applications). The Pi 3 also supports wireless internet, with built-in Wi-Fi and Bluetooth. The latest board can also boot directly from a USB-attached hard drive or pen drive, supports booting from a network-attached file system and supports the use of PXE (which is useful for remotely updating a Pi and for sharing an operating system image between multiple machines). [15.] In this project, the Raspberry Pi's primary use is to read the data from the MCP3008 (through its channels), convert it into EC and pH values and display the results to the users. The Raspberry Pi is also connected to two red LEDs that turn on when the pH and EC are in unacceptable ranges for proper plant growth.

#### 4.1.1 Circuit diagram

The pH electrode (shown on the top left side of Figure 9) is connected to the pH meter via BNC cable. Depending on the pH value of a specific substance, the meter will receive voltage values between 414.12mV and -414.12mV as an input from the electrode. The pH meter has four connection points in need of routing, each of which are extremely necessary to make the system work. After connecting the electrode to the meter, three connections are left to be made (VCC, Vout and GND). Voltage common collector (VCC for short) powers the meter and is connected via red cable to a 3.3V voltage source. The ground (GND) pin is connected via black cable to a ground pin on the Raspberry Pi. The output voltage (Vout) is connected to a power isolation circuit via brown cable. As mentioned in the previous section, the isolation circuit contains a summing amplifier, an LED, an LDR, and two inverting (gain) amplifiers. To mirror the circuit shown in Figure 8 in real life, LM386 is used in place of operational amplifier A1, NSL-32R3 is used in place of the LED and LDR and LM358 is used in place of inverting amplifiers A2. The isolation circuit for the pH meter is shown on the second half of the full sized breadboard (in Figure 9), after the dotted neon pink line. The output of this circuit is connected via brown cable to channel 1 of the MCP3008.

The EC electrode (shown on the bottom left side of Figure 9) is connected to the EC meter via BNC cable. Depending on the electrical conductivity value of a specific substance, the meter will receive small voltage values close together in value as an input from the electrode. The EC meter (like its pH counterpart) has four connection points in need of routing, each of which are completely necessary to make the system work. After connecting the electrode to the meter, three connections are left to be made (VCC, Vout and GND). Voltage common collector (VCC for short) powers the meter and is connected via red cable to a 3.3V voltage source. The ground (GND) pin is connected via black cable to a ground pin on the Raspberry Pi. The output voltage (Vout) is connected to a power isolation circuit via brown cable. As mentioned in the previous section, the isolation circuit contains a summing amplifier, an LED, an LDR, and two inverting (gain) amplifiers. To mirror the circuit shown in Figure 8 in real life, LM386 is used in place of operational amplifier A1, NSL-32R3 is used in place of LED D1 and LDR R6 and LM358 is used in place of gain amplifier A2. The isolation circuit for the EC meter is on the first half of the full sized breadboard (in Figure 9), before the dotted neon pink line. The output of this circuit is connected via brown cable to channel 0 of the MCP3008.

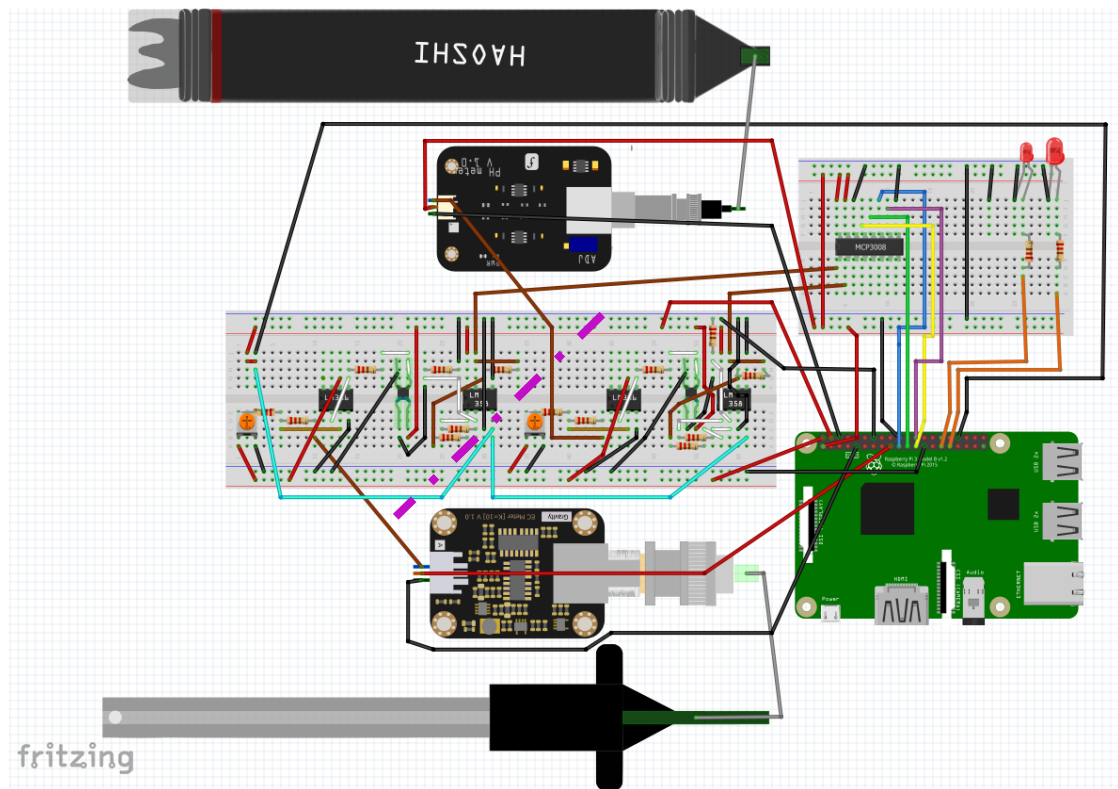


Figure 9. A circuit diagram displaying all the components used in the project.

The next step is to connect the MCP008 to the Raspberry Pi 3. The voltage drain drain (VDD for short) and the Voltage reference (VREF) of the ADC are connected via red cable to the 3V3 common voltage reference, while the ground (AGND and DGND) pins are connected via black cable to the common ground. The CLK pin of the MCP3008 is connected to the SCLK pin of the Raspberry Pi (via yellow cable) in order to synchronise their clock signals, the chip select (CS) pin of the MCP3008 is connected to the CE0 pin of the Raspberry Pi (via purple cable) to prepare the ADC for an SPI connection between the two devices, the data out (DOUT) pin of the MCP3008 is connected to the master in slave out (MISO) pin of the Raspberry Pi (via green cable) and is used by the Pi to receive data from the ADC. The data in (DIN) pin of the MCP3008 is connected to the master out slave in (MOSI) pin of the Raspberry Pi (via blue cable), and is used to receive data from the microcontroller. Finally, one red LED is connected to GPIO5 and the other to GPIO6. These two LED's sole purpose is to serve as indicators that the EC and pH values of the solutions being read are in the acceptable ranges. If the pH is not within the correct limits, LED 2 (In Figure 9, on the far right of the breadboard) turns on and if the EC is not within the correct limits, LED 1 turns on.

### 4.1.2 Raspberry Pi 3 configuration

The Raspberry Pi needs to be configured to turn on the warning lights (when the EC and pH values are in unacceptable ranges) and to accept values from the ADC, before converting them to the required quantities. The section of code required to facilitate this process is shown in Listing 1. The first step in the configuration process is to set pin 29 and 31 (of the Pi) as output pins. The initial values of these pins have to be set to low to prevent the LED's from turning on without being explicitly prompted to do so.

```
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(29, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(31, GPIO.OUT, initial=GPIO.LOW)
# Set pin 29 and 31 to be output pins and set their initial values to low
(off)

ec      = DFRobot_EC()
ph      = DFRobot_PH()

#ph.reset()

ec.begin()
ph.begin()

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz=1000000

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
    adc = spi.xfer2([1, (8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts

# Define sensor channels
EC_channel = 0
PH_channel = 1

# Define delay between readings
delay = 5
```

Listing 1. An excerpt from the SPI connection, function definition and GPIO pin setup section of the main the Raspberry Pi code [25]. The code in its entirety is shown in Appendix 1.

The next step is to open an SPI bus to permit short distance communication between the microcontroller and the MCP3008. The data present in the 8 channels of the ADC can be acquired through this link and the reading process can be initiated by calling the

ReadChannel function. Afterwards, the channels as to which the sensors are connected to are defined (the EC meter is linked to channel 0 and the pH meter is linked to channel 1) and a delay of 5 seconds is added so as not to overload the system.

```
while True:

    temperature = 25
    # Read the EC sensor data
    EC_level = ReadChannel(EC_channel)    cur.execute(sql)

    # Convert EC sensor data to voltage level
    EC_volts = ConvertVolts(EC_level,2)
    EC_millivolts = EC_volts * 1000 #Convert EC value from V to mV

    #Convert voltage to EC with temperature compensation
    EC_mS_per_centimeter = ec.readEC(EC_millivolts,temperature)
    EC = EC_mS_per_centimeter * 100

    # Read the PH sensor data
    PH_level = ReadChannel(PH_channel)

    # Convert PH sensor data to voltage level
    PH_volts = ConvertVolts(PH_level,2)
    PH_millivolts = PH_volts * 1000

    #Convert voltage to PH with temperature compensation
    PH = ph.readPH(PH_millivolts,temperature)

    # Print out results
    print ("-----")
    print ("EC:{}".format(EC))
    print ("pH:{}".format(PH))

    if EC > 250 or EC < 150:
        print("The EC is beyond acceptable limits!")
        GPIO.output(29, GPIO.HIGH) # Turn on

    else:
        GPIO.output(29, GPIO.LOW) # Turn off

    if PH > 6.9 or PH < 6.7:
        print("The pH is beyond acceptable limits!")
        GPIO.output(31, GPIO.HIGH) # Turn on led

    else:
        GPIO.output(31, GPIO.LOW) # Turn off led

    # Wait before repeating loop
    time.sleep(delay)
```

Listing 2. An excerpt from the while loop of the main the Raspberry Pi code. The code in its entirety is shown in Appendix 1.

A while loop (shown in Listing 2) is then subsequently initiated. At this point, the ReadChannel function is called and the data in channel 0 (of the Raspberry Pi) is read by the system. The information in this channel comes from the ADC, whose output is a range of numbers between 0 and 1023 (a reading of 0 means the output is 0V and a reading

of 1023 means the output is 3.3V). This data is converted to voltage by the use of the previously defined ConvertVolts function and stored in the variable EC\_volts. In its current state, this value is incompatible with the readEC function of the DFRobot\_EC.py file (Appendix 2). To amend this, the voltage value stored in EC\_volts is converted to millivolts (by multiplying it by 1000) and saved in the variable EC\_millivolts. The readEC function can then be called and used to convert the millivolt value to an EC value, while compensating for the ambient temperature. Since a majority of the measurements need to be done in a controlled setting (at room temperature), the value of temperature is set to 25. The unit of measurement chosen for EC in this project is millisiemens per meter (mS/m). The DFRobot\_EC.py file (Appendix 2) converts values in millivolts to electrical conductivity values in millisiemens per centimeter (mS/cm) and stores them in the variable EC\_mS\_per\_centimeter. To convert the values from mS/cm to mS/m, the data in EC\_mS\_per\_centimeter is multiplied by 100 and saved to the variable EC. The formula that the DFRobot\_EC.py file uses to convert voltage values to EC values is given in the equation (11) below.

$$EC = \frac{100 \cdot EC \text{ meter } V_{out} \cdot \text{Cell constant}}{820 \cdot 200} \quad (11)$$

The ReadChannel function is then called upon once again to read the data in channel 1 of the Raspberry Pi. This data is converted to voltage by the use of the ConvertVolts function and stored in the variable PH\_volts. In this state, this value is incompatible with the readPH function of the DFRobot\_PH.py file (Appendix 2). To fix this, the voltage value stored in PH\_volts is converted to millivolts (by multiplying it by 1000) and saved in the variable PH\_millivolts. The readPH function can then be called and used to convert the millivolt value to a pH value, while compensating for the ambient temperature [16]. The formula that the DFRobot\_PH.py file uses to convert voltage values to pH values is given in the equation (12) below.

$$pH = \left( \left( \frac{7-4}{pH \text{ meter } V_{out(pH 7)} - pH \text{ meter } V_{out(pH 4)}} \right)^{7-4} \cdot (pH \text{ meter } V_{out} - pH \text{ meter } V_{out(pH 7)}) \right) \cdot 7 \quad (12)$$

The next step in this process is to print out the values of EC and pH obtained to the screen, so that the users of the system can keep track the sensor data. If the value of electrical conductivity detected is above 250 or below 150, pin 29 is set to high and the LED attached to the pin turns on. Similarly, if the value of pH detected is above 6.9 or below 6.7, pin 31 is set to high and the LED attached to the pin turns on.



### 4.1.3 Sensor calibration

Default values for EC and pH (that correspond to specific meter voltages) already exist in the system. However, if at any point the sensors become less accurate and need to be recalibrated it can be done with the sections of code specified in Listings 3 and 4. The code shown in Listing 3 is used for EC sensor calibration. For the entire process to begin, the EC electrode has to be placed in a buffer solution and the `DFRobot_EC_Calibration.py` file (shown in Appendix 3) has to be run on the Raspberry Pi device. Much like the code for the EC and pH measurements (shown in Appendix 1), this code makes use of the `DFRobot_EC.py` file.

The Raspberry Pi connects to the MCP3008 by means of an SPI connection, reads the data in channel 0 using the `ReadChannel` function, converts the channel data to voltage level, then calls the calibration function in the `DFRobot_EC.py` file. The calibration function calculates a raw EC value using the same formula as specified in equation (11), but without multiplying the value with the cell constant.

If the EC electrode is placed in a buffer solution with a conductivity value of 1.413mS/cm, then the raw EC value calculated will be between 0.9mS/cm and 1.9mS/cm. On the other hand, if the EC electrode is placed in a buffer solution with a conductivity value of 12.88mS/cm, then the raw EC value calculated will be between 9mS/cm and 16.8mS/cm. This value is used to calculate the cell constant of the sensor and the result is saved in `ecdata.txt`. From this moment forward, every time the `readEC` function is called (whilst running the main code) the cell constant calculated in the calibration process will be used in determining the value of the electrical conductivity. If the cell constant value obtained is proven to be incorrect and the user wants to use the default value, this can be achieved by uncommenting the `ec.reset()` line in the main measurement code. This command will reset the value of the cell constant to its default value of 10.

```

while True:

    temperature = 25

    # Read the EC sensor data
    EC_level = ReadChannel(EC_channel)

    # Convert EC sensor data to voltage level
    EC_volts = ConvertVolts(EC_level,2)
    EC_millivolts = EC_volts * 1000 #Convert EC value from V to mV

    print("CH0: {}mV".format(EC_millivolts))

    #Calibrate the calibration data
    ec.calibration(EC_millivolts)

    # Wait before repeating loop
    time.sleep(1.0)

```

Listing 3. An Excerpt from the EC sensor calibration code in the Raspberry Pi.

The code shown in Listing 4 is used for pH sensor calibration. For the entire process to begin, the pH electrode has to be placed in a pH buffer solution and the DFRobot\_PH\_Calibration.py file (shown in Appendix 3) has to be run on the Raspberry Pi device. Much like the code for the EC and pH measurements (shown in Appendix 1), this code makes use of the DFRobot\_PH.py file.

The Raspberry Pi connects to the MCP3008 by means of an SPI connection, reads the data in channel 0 using the ReadChannel function, converts the channel data to voltage level, then calls the calibration function in the DFRobot\_PH.py file. A fully functional pH sensor from DFRobot will produce a voltage between 1322mV and 1678mV when the pH electrode is placed in a buffer solution with a pH value of 7. On the other hand, when the sensor is placed in a buffer solution with a pH value of 4, it will produce a voltage between 1854 mV and 2210mV. The value produced by the sensor at either of these two pH values will be saved to the file phdata.txt. From this moment forward, every time the readPH function is called (whilst running the main code) the voltage values recorded in the calibration process will be used in determining the value of the pH. The voltage value obtained while placing the pH electrode in the solution of pH 7 will be used instead of pH meter  $V_{out(pH\ 7)}$  and the voltage value obtained while placing the pH electrode in the solution of pH 4 will be used instead of pH meter  $V_{out(pH\ 4)}$  in equation (12). If the voltage values obtained are proven to be incorrect and the user wants to use the default values, this can be achieved by uncommenting the ph.reset() line in the main measurement code. This command will reset the variable pH meter  $V_{out(pH\ 7)}$  to 1500mV and the variable pH meter  $V_{out(pH\ 4)}$  to 2032.44mV.

```
while True:

    temperature = 25

    # Read the PH sensor data
    PH_level = ReadChannel(PH_channel)

    # Convert PH sensor data to voltage level
    PH_volts = ConvertVolts(PH_level,2)
    PH_millivolts = PH_volts * 1000

    print("CH1: {}mV ".format(PH_millivolts))

    #Calibrate the calibration data
    ph.calibration(PH_millivolts)

    # Wait before repeating loop
    time.sleep(1.0)
```

Listing 4. An Excerpt from the pH sensor calibration code in the Raspberry Pi.

The code shown in Listing 3 only allows calibration using buffer solutions with an EC of 141.3mS/m or 1288mS/m; any other solutions will be ignored by the program. Similarly, the code shown in Listing 4 will only allow pH sensor calibration using buffer solutions with a pH of 7 or 4 and ignore all others. This is because these are the only calibration solutions offered by DFRobot. Since this company made the sensing meters and the buffer solutions, they know what voltages correspond to what EC and pH values and have posted this information on their website. This was the data that was used to create the DFRobot\_EC.py and the DFRobot\_PH.py files. If the need for calibrating the system using buffer solutions that are not previously mentioned arises, the user would have to alter the source codes of these two files to account for the new calibration solutions.

#### 4.1.4 Circuit Simulation

Proteus is a design software tool invented by Labcenter Electronics to design, draw, and simulate circuits in real time. It gives its users the ability to make two-dimensional and three-dimensional circuit designs. With the use of this software, different electrical and electronic circuits can be simulated on laptops or personal computers. It has large variety of components in its library. It has oscilloscopes and voltmeters (for measurement and analysis), probes for real time monitoring of the parameters of a circuit, voltage and current sources, signal generators, analogue and digital Integrated circuits, resistors, semiconductor switches, microcontrollers and many more. [24.]

In this project Proteus was used to simulate the operation of the overall circuit. A small snippet of the simulation is shown on Figure 10. The Raspberry Pi 3 device is represented by RPI3 (U1) and the warning lights (which are meant to turn on when the EC and pH values are outside the acceptable range) are represented by LED's D1 and D2 respectively. The MCP3008 is represented by U2, the output of the isolation circuit for the EC sensor is represented by OPTO-Isolator(EC) and finally the output of the isolation circuit for the pH sensor is represented by OPTO-Isolator(pH). Although Proteus allows its users to program Pi devices using flowchart blocks, RPI3 was programmed in python in order to mimic a real life scenario. Apart from the basic python libraries, Proteus only supports the smbus, pygame, wiringpi, RPi.GPIO, and spidev libraries. In order to use other libraries, they have to be downloaded and saved in a folder, then a link has to be established between Proteus and the file location using the sys.path.insert command.

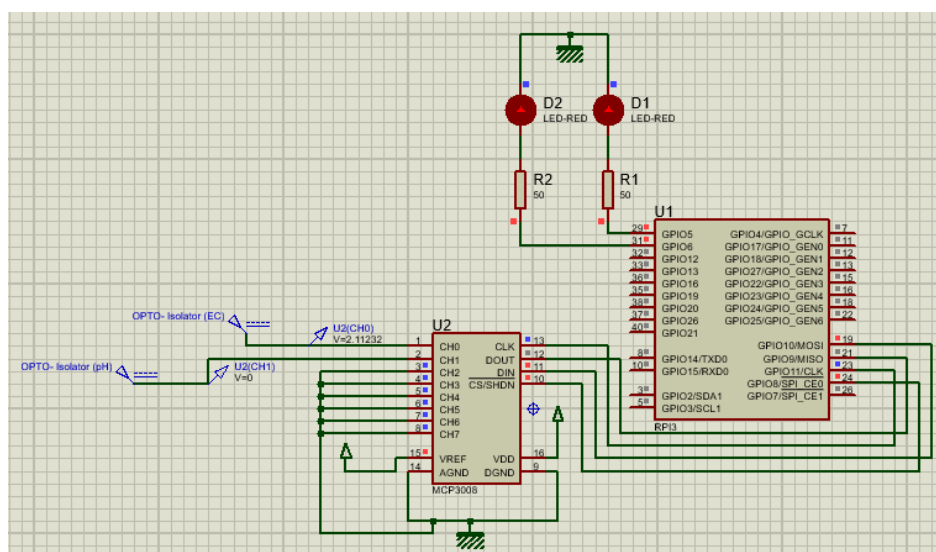


Figure 10. A snapshot of the circuit simulation on Proteus.

The voltages going to the MCP3008 from the power isolation circuits of the EC meter and the PH meter are shown by probes U2(CH0) and U2(CH1) respectively. The voltage from the EC sensor in the simulation above is 2.112V and the voltage from the pH sensor is 0V.

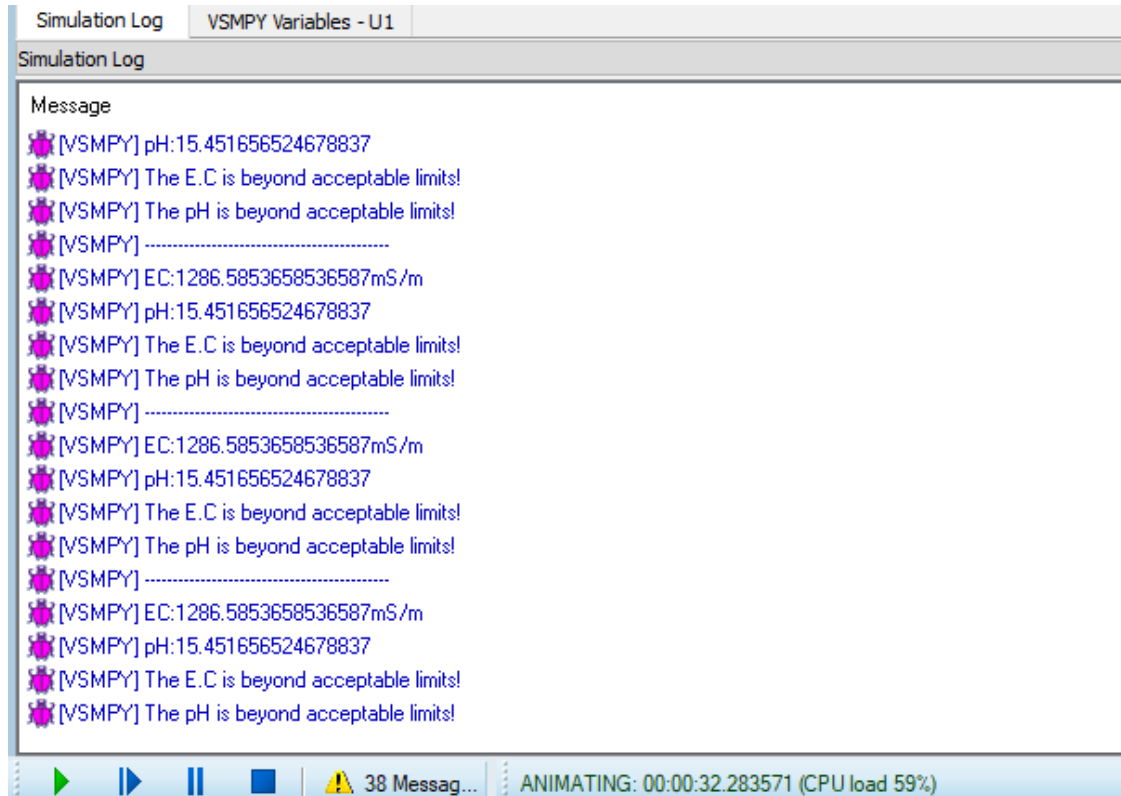


Figure 11. The output of the simulation.

Figure 11 shows the output of the simulation. When the output of the pH sensor is 0V, the pH value calculated by the program is approximately 15.245 and when the output of the EC sensor is 2.112V, the EC value calculated by the program is approximately 1286.59mS/m. Both these values are outside the acceptable range for growing healthy Hops plants; the simulated Raspberry Pi device outputs warning messages to its users saying “The E.C is beyond the acceptable limits!” and “The pH is beyond the acceptable limits!”. RPI3 will make LED D1 turn on if the EC value calculated is outside the optimal range and make LED D2 turn on if the pH value calculated is outside the optimal range. In this case, both values are outside the suitable range and so both D1 and D2 turn on.

#### 4.1.5 Results

The data collected by comparing the components used in this thesis to devices produced by well-known manufacturers (in addition to the simulation outcomes) heavily supports the theory that under the right series of circumstances, inexpensive pH and EC measurements are entirely possible. The biggest piece of evidence that supports the aforementioned claim is the data presented in Table 4.

Table 4. A table comparing the measurement circuit system used in this thesis to one created using AtlasScientific™ devices.

Cost effective measurement system			AtlasScientific™		
Part type	Accuracy	Price(€)	Part type	Accuracy	Price(€)
DFRobot Meter Pro Kit ( pH electrode + pH meter )	± 0.1pH	49.78	pH Kit (EZO™ pH Circuit + pH Probe + calibration solutions + pH storage solution + Electrically Isolated EZO™ Carrier Board)	± 0.002pH	149.49
DFRobot Analog Electrical Conductivity Sensor kit with K=10 (EC meter + EC laboratory probe+ buffer solution)	± 5%	69.90	Conductivity Kit with K= 10 (EZO™ Conductivity Circuit + Conductivity Probe + calibration solutions )	± 2%	217.74
Opto-Isolators	-	6	Basic EZO™ In-line Voltage Isolator	-	23.69
Raspberry Pi 3 Official Starter Kit	-	64.50	Raspberry Pi 3 Official Starter Kit	-	64.50
Alvin instruments EC electrode	± 5%	40.46	-	-	-
MCP3008	±0.1	1.52	-	-	-
<b>Total price</b>		<b>232.16</b>	<b>Total price</b>		<b>455.42</b>

Atlas Scientific is a well-known company that focuses on manufacturing laboratory and industrial grade sensing instrumentation for robots, appliances and industrial management systems. Numerous Atlas Scientific sensors can be found within a comprehensive selection of merchandise and devices all around the globe. From mission-critical military applications to Arduino based university projects, their line of business serves to make

any device thinkable incredibly accurate. That being said, their products are more expensive than the ones made by DFRobot and Alvin Instruments. A measurement system constructed using AtlasScientific™ devices would cost nearly double the price of the cost effective measurement system. In the context of maintaining the EC and pH of the watering solution for Hops plants, the cheaper option of the two would work just as well as its more expensive counterpart.

Table 5. The output voltages of the pH electrode and pH meter with respect to the pH of the substance being measured.

pH	pH electrode output voltage (V)	pH meter output voltage (V)
0.0	0.414120	2.74236
1.0	0.354960	2.56488
2.0	0.295800	2.38740
3.0	0.236640	2.20992
4.0	0.177480	2.03244
5.0	0.118320	1.85496
6.0	0.059160	1.67748
6.6	0.035496	1.57099
7.0	0.000000	1.50000
8.0	-0.059160	1.32252
9.0	-0.118320	1.14504
10.0	-0.177480	0.96756
11.0	-0.236640	0.79008
12.0	-0.295800	0.61260
13.0	-0.354960	0.43512
14.0	-0.414120	0.25764

The DFRobot pH sensor is less accurate than its more high-priced equivalent (the accuracy of the DFRobot pH sensor is  $\pm 0.1\text{pH}$  and the accuracy of the AtlasScientific™ pH sensor is  $\pm 0.002\text{pH}$ ). The results acquired by simulating the pH section of the circuit and interpreting the data produced are shown in Table 5. The data indicates that consecutive output voltages of the meter (corresponding to different pH values) differ by a very small value. However, this does not have a very significant effect on the perceived accuracy of the pH measurements. In the grand scheme of things, if the pH of the watering solution is 0.1pH greater or smaller than intended, it would not negatively impact the growth of Hops plants to an extreme degree.

Table 6. The output voltages of the EC meter with respect to the EC of the substance being measured.

Electrical Conductivity (mS/m)	EC meter output voltage (V)
0.0	0.000000
100.0	0.164000
141.3	0.231732
150.0	0.246000
200.0	0.328000
250.0	0.410000
270.0	0.442800
1288.0	2.112320
2000.0	3.280000

The DFRobot electrical conductivity sensor is also less accurate than its more expensive alternative (the accuracy of the DFRobot electrical conductivity sensor is  $\pm 0.5\text{mS/m}$  and the accuracy of the AtlasScientific™ electrical conductivity sensor is  $\pm 0.2\text{mS/m}$ ). The results obtained by simulating the EC subdivision of the circuit and interpreting the data produced are shown in Table 6. Each consecutive EC meter output voltage differs from the value that precedes it by a small amount, meaning that even a small measurement error would have a significant impact on what the Raspberry Pi perceives to be the EC value. This is the one of the downsides of the cost effective system; when trying to maintain the perfect EC for keeping plants alive it is better to be more accurate than less.



## 4.2 Mindsphere

MindSphere is an open, cloud-based IoT OS created by Siemens. It facilitates data collection in real time and provides its users access through most browsers. It also allows its users to establish secure links to the devices being monitored and perform predictive maintenance [18]. Of the numerous plans currently being offered by Siemens, Metropolia University of Applied Sciences has opted to acquire the MindAccess DevOps plan. This plan is specifically made for developers i.e. it permits the use of the pre-packaged applications readily available on MindSphere and allows applications built elsewhere to be added to the platform.

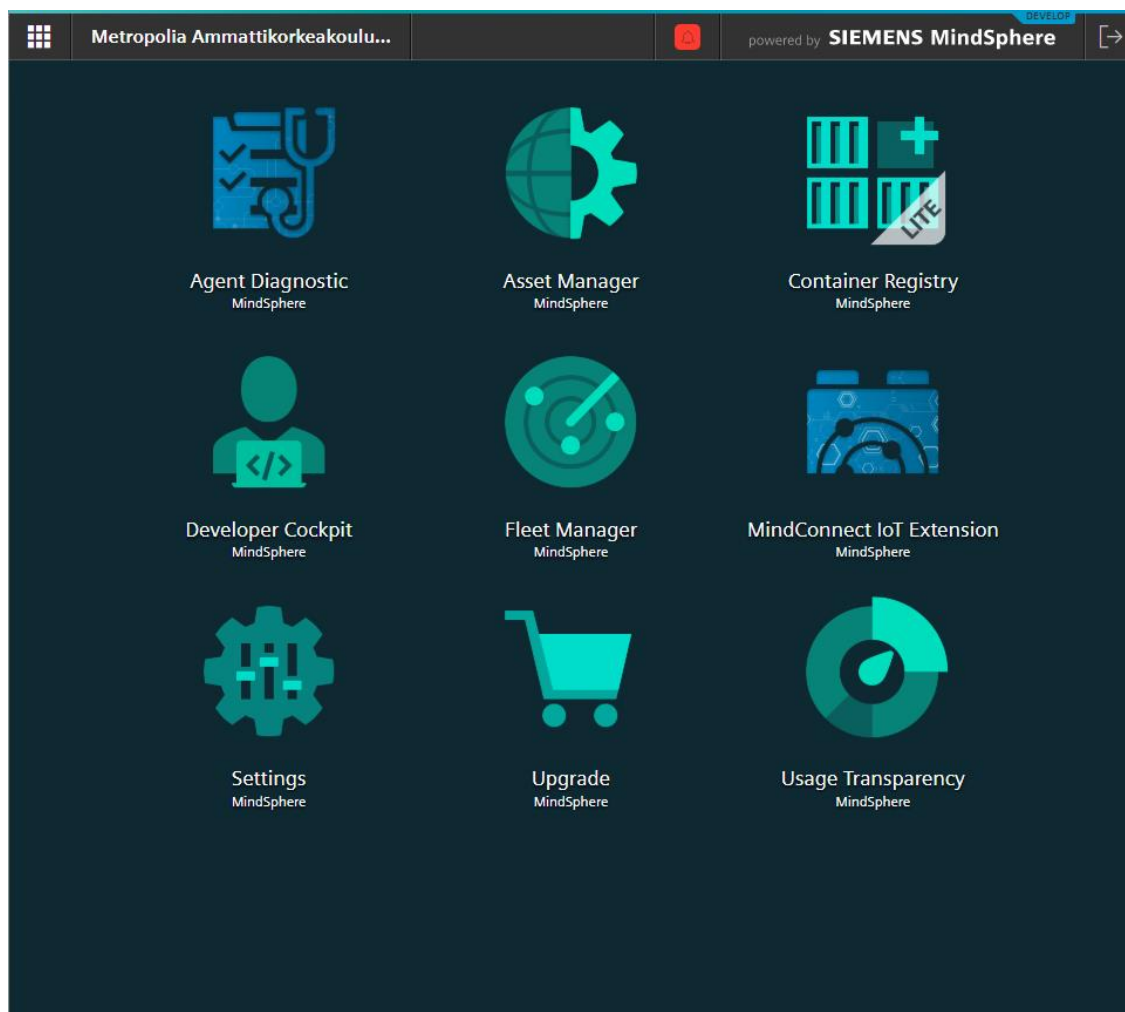


Figure 12. An excerpt from MindSphere's launchpad.

As seen in Figure 12, the MindAccess DevOps plan has nine key applications and features. The Agent Diagnostic is an application that provides a user interface for creating agent monitors and gaining insights into already existing agent monitors (such as the MindConnect Nano). It is built on the diagnostic feature of the MindConnect API, allows the activation and deactivation of agents and allows the system's admins to view their messages. The Asset Manager application lets its users simulate the complete structure of an industrial process by connecting assets and creating data models for how specific asset data is evaluated. The Fleet Manager application offers its users monitoring, management, visualization and alerts capabilities. The alerts allow the users of the application to locate individual assets and their configuration details, monitor key metrics and create rules that will send alerts if they become triggered. [22.]

The Container Registry application allows its users to fast-track development and streamline the storage and management of images. With the use of the aforementioned application, users can view a project and its details, create "Robot Accounts" (accounts intended to perform docker push / docker pull operations using a token), tag and push images to the Container Registry as well as allowing its users to view the log information of the operations performed in the application. The Developer Cockpit aids developers in configuring and managing their developed applications in MindSphere. The Usage Transparency Service collects numerous consumption metrics in order to allow its users to view the usage details of MindSphere applications for a specific time period, and to generate usage reports. The Upgrade feature gives MindSphere customers the opportunity to purchase upgrades and use them instantly on their MindAccess Plan account. With Settings, one can manage users and their permissions, add company specific information to the tenant (a representation of a real-world organization on MindSphere) and create subtenants (a certain limited resource of a tenant which represents a subpart of a real-world organization). [22.]

Lastly, there is the MindConnect IoT Extension, an incorporation layer that supports various conventions and permits nearly any IoT-Ready gadget (from any seller) to connect to MindSphere. Furthermore, the MindConnect API facilitates the programming of connectivity agents for a predetermined use. [22.]

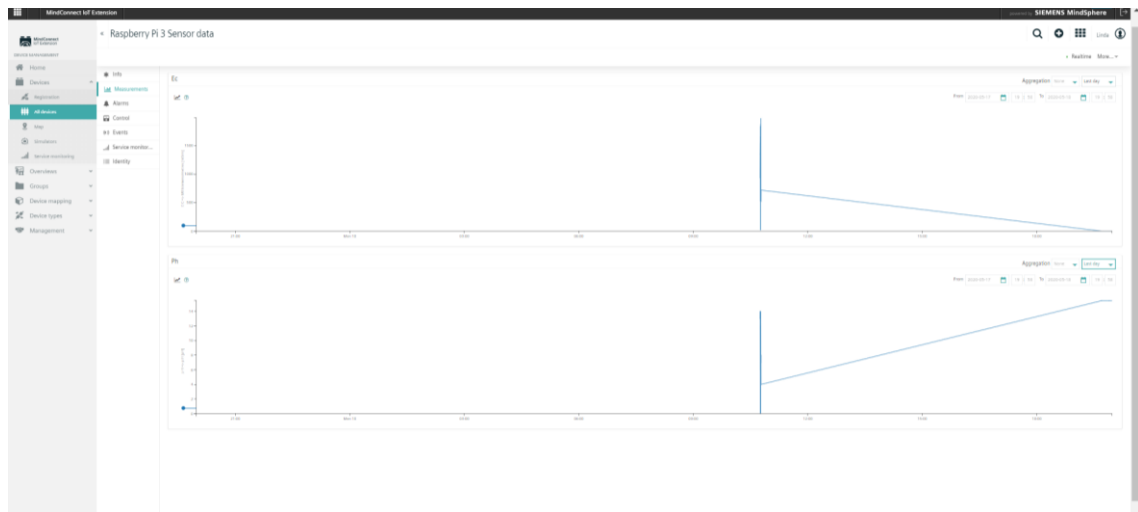


Figure 13. An excerpt from MindSphere's IoT extension.

In this project, the IoT extension of MindSphere (MindConnect) was used to monitor and graph sensor data (the graphs are seen in Figure 13). The extension also had the added benefit of allowing real time data synchronization and measurement analysis via the Data Explorer.

```
mqtt.Client.connected_flag=False #create flag in class
client = mqtt.Client(clientId)
client.username_pw_set(tenant + "/" + username, password)
client.on_message = on_message
client.on_publish = on_publish

# connect the client to MindConnect IoT
client.on_connect=on_connect #bind call back function
client.loop_start()
client.connect(serverUrl)

#create device (Raspberry Pi 3)
publish("s/us", "100," + device_name + ",MCIoT_MQTTdevice", True)
#set required connection interval to 10 minutes
publish("s/us", "117,10")

#Send Data via MQTT
publish("s/us", "110,000000008b64f58a,RaspPi BCM2835,a02082")
publish("s/us", "114,MCIoT_Restart")
print("Device registered successfully!")

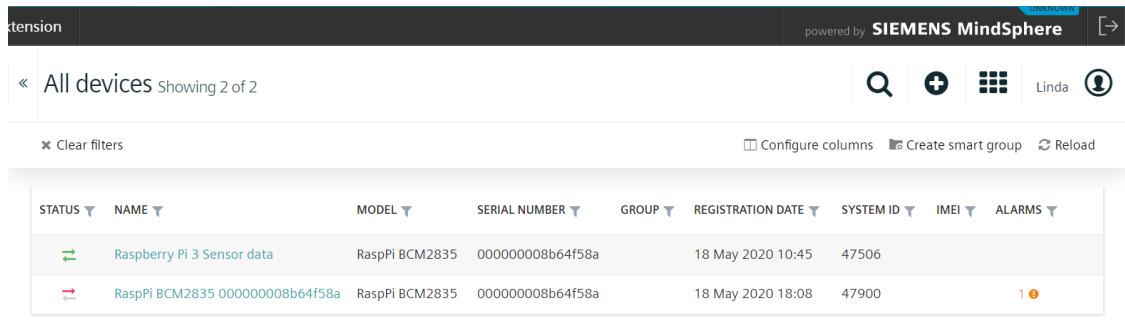
client.subscribe("s/ds")
sendMeasurements()
```

Listing 5. The section of code in the Raspberry Pi that connects and sends the sensor values to the MindConnect IoT extension.

MindConnect is incapable of reading the data being inputted and outputted by the Raspberry Pi without being explicitly prompted to do so. Short of using devices like the Master Brick 2.1 (by Tinkerforge), one of the only ways to relay the information on the Raspberry Pi to MindSphere is to do so with the use of an MQTT connection[17]. Message Queue Telemetry Transport (MQTT for short) is a messaging transport protocol, which uses the brokered publish/subscribe system. This configuration separates the publisher client (who sends a particular message) from the subscriber client (who receives it). The code written in Listing 5 shows how the MQTT client/server connection (between MindConnect and the Pi) was formed and utilised [21].

Firstly, the Raspberry Pi 3 attempted to establish a secure link to the URL of the IoT platform using the connect acknowledge flag, `mqtt.Client.connected_flag`. This flag returns a code that tells the client (which is the Raspberry Pi 3 in this scenario) whether the connection attempt was successful or not. There are a total of six return codes that the flag can return. If the flag returns a code with the value 0, then the connection is accepted by the MQTT broker (which is the MindConnect extension in this case). If the flag returns a code with the value 1, then the connection is refused due to an unacceptable protocol version. If the flag returns a code with the value 2, then the connection is refused due to a rejected identifier. If the flag returns a code with the value 3, then the connection is refused due to the server being unavailable. If the flag returns a code with the value 4, then the connection is refused due to a bad username or password. Finally, If the flag returns a code with the value 5, then the connection is refused because the client trying to access the broker is not authorized to do so.

After a successful connection to the server is made, a Raspberry Pi 3 device is created on the platform using the `publish("s/us", "100," + device_name + ",MCIoT_MQTTdevice", True)` command. The name of the device on the platform is represented by the variable `device_name` and the type of device is represented by `MCIoT_MQTTdevice`. As seen in Figure 14, the name of the device chosen by user was Raspberry Pi 3 Sensor data.



powered by **SIEMENS MindSphere**

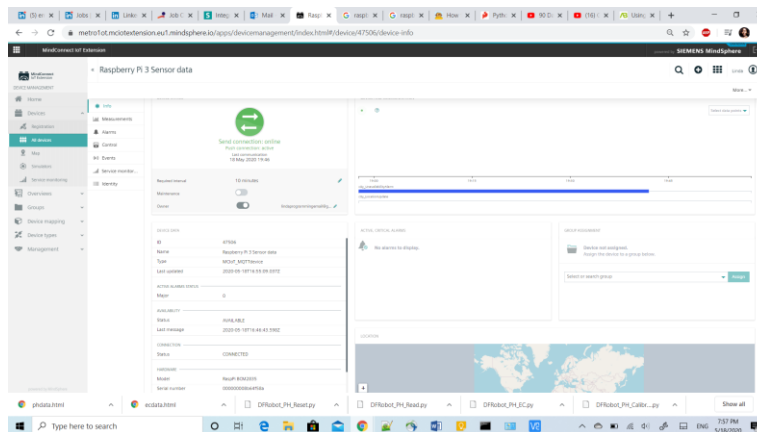
All devices Showing 2 of 2

Clear filters Configure columns Create smart group Reload

STATUS	NAME	MODEL	SERIAL NUMBER	GROUP	REGISTRATION DATE	SYSTEM ID	IMEI	ALARMS
	Raspberry Pi 3 Sensor data	RaspPi BCM2835	000000008b64f58a		18 May 2020 10:45	47506		
	RaspPi BCM2835 000000008b64f58a	RaspPi BCM2835	000000008b64f58a		18 May 2020 18:08	47900		1

Figure 14. An excerpt from MindSphere's IoT extension showing all the registered devices.

To increase the time interval between measurements, the `publish("s/us", "117,10")` command is used. In this command, 10 represents the amount of minutes the user wants the connection to remain open. Changing this number will change the amount of time the connection will stay open. The serial number, hardware model and revision of the device connecting to the platform can be set using the command `publish("s/us", "110,000000008b64f58a,RaspPi BCM2835,a02082")`. As seen in Figure 15, the serial number of the Raspberry Pi device was set to 000000008b64f58a, the model of the device was set to RaspPi BCM2835 and the revision of the device was set to a02082 [20, 2].



Raspberry Pi 3 Sensor data

Send connection status  
Registration date  
18 May 2020 10:45

Refresh interval: 10 minutes

Measurements:

Device:

Device ID: 47506  
 Model: RaspPi BCM2835  
 Type: RaspPi BCM2835  
 Last update: 2020-05-18T18:08:00Z

Active System ID: 47506

Model: RaspPi BCM2835  
 Revision: a02082

Device ID: 47506  
 Status: CONNECTED

IMEI: 000000008b64f58a  
 Serial number: 000000008b64f58a

Location:

Figure 15. An excerpt from MindSphere's IoT extension.

After initiating an automatic refresh of the IoT platform, the Raspberry Pi listened for built-in commands and sent the EC and pH values obtained to the MindConnect child device using the `sendMeasurements()` command.

## 5 Conclusion

Oftentimes, it is thought that the accurate measurement of sensor data using industrial grade equipment costs a significant amount (no matter what device manufacturer is chosen). This line of thinking is prevalent due to the irrational thought amongst consumers that expensive items are better in quality than their cheaper counterparts [19]. This thesis however, proves that cost effective EC and pH measurements are possible if done in situations where the precision of the measurements is not a mitigating factor.

Two groups in particular will be impacted the most by the findings of this thesis, individuals with small at home gardens and small farmers with businesses that operate locally. They can use the information presented in this document as a guideline for creating an alternative system for data measurement that rivals the ones currently in the open market in quality, but made at a significantly lower price. Large companies often have long standing relationships with other big corporations and usually get the equipment they deem necessary at a much lower cost than the retail value. Due to this reason, they will be affected the least by the discoveries made in this thesis.

A study investigating the significance of using inexpensive microcontrollers with readily available analogue input pins (e.g. ones in the Arduino family) in a system similar to the one presented in this thesis could be a great area of future research. Such embedded controllers would have the added benefit of saving the users of the measurement system some money, since there would be no need for an ADC in the circuit. Subsequently, the measurements may also slightly increase in accuracy because the power loss of the overall system would decrease. Another great field of study would be one that investigates whether or not the circuit presented in this thesis is capable of measuring values from other sensors in addition to the ones present originally. Such a study would help consumers know if the system can perform multiple measurement tasks without unexpectedly giving out.

In conclusion, the data obtained from this thesis proves that money is not an obstacle to creating a fairly accurate system for measuring sensor data. With enough time and effort, the current system can be used for a large variety of applications (both in the agricultural industry and beyond).

## References

- 1 How to Grow Fresh Hydroponic Hops – MaximumYield Inc [Internet]. Maximumyield.com. 2020 [cited 28 May 2019]. Available from: <https://www.maximumyield.com/how-to-grow-fresh-hydroponic-hops/2/17593>
- 2 HydroHumala – Redono Oy [Internet]. Redono.fi. 2017 [cited 16 May 2019]. Available from: <https://www.redono.fi/products/hydrohumala/>
- 3 Water Conductivity – Lenntech B.V. [Internet]. Lenntech.com. 1998-2020 [cited 26 September 2019]. Available from: <https://www.lenntech.com/applications/ultrapure/conductivity/water-conductivity.htm>
- 4 Water quality in home gardens – Government of Western Australia [Internet]. Agric.wa.gov.au [cited 11 December 2014]. Available from: <https://www.agric.wa.gov.au/water-quality-home-gardens?nopaging=1>
- 5 Cascade Hops – Edible Landscaping [Internet]. EdibleLandscaping.com. [cited 17 April 2019]. Available from: <http://ediblelandscaping.com/products/herbs/Hops/CascadeHops.php>
- 6 How to Grow Hops At Home – AMERICAN HOMEBREWERS ASSOCIATION [Internet]. Homebrewersassociation.org 2020 [cited 4 November 2018]. Available from: <https://www.homebrewersassociation.org/how-to-brew/how-to-grow-hops/>
- 7 Fusarium Infections in Immunocompromised Patients – American Society for Microbiology [Internet]. Cmr.asm.org 2020 [cited 21 April 2020]. Available from: <https://cmr.asm.org/content/20/4/695>
- 8 Nutrient Management and Imbalances [Internet]. David H. Gent, J Robert Sirrine, and Heather M. Darby; 2015 [cited 2015]. Available from: <https://www.usahops.org/cabinet/data/9.pdf>
- 9 Gravity: Analog Electrical Conductivity Sensor / Meter (K=10) – DFRobot [Internet]. Dfrobot.com 2008 [cited 19 May 2019.]. Available from: <https://www.dfrobot.com/product-1797.html>
- 10 Gravity: Analog pH Sensor / Meter Pro Kit For Arduino – DFRobot [Internet]. Dfrobot.com 2008 [cited 19 May 2019]. Available from: <https://www.dfrobot.com/product-1110.html>
- 11 Industrial Online Conductivity Electrode probe sensor 316L stainless cell constant 0.01 0.1 1 10 – Alvin Instrument [Internet]. Aliexpress.com 2010 - 2020. Available from: <https://www.aliexpress.com/item/32949252890.html?spm=a2g0o.cart.0.0.1ff83c00huqZ9U&mp=1>

- 12 DFRobot Gravity: analog electrical conductivity meter V2 – DFRobot [Internet]. Dfrobot.com 2008 [cited 19 May 2019]. Available from: [https://wiki.dfrobot.com/Gravity\\_Analog\\_Electrical\\_Conductivity\\_Sensor\\_Meter\\_V2\\_K=1\\_SKU\\_DFR0300](https://wiki.dfrobot.com/Gravity_Analog_Electrical_Conductivity_Sensor_Meter_V2_K=1_SKU_DFR0300)
- 13 A pH scale on white background – Eezy Inc [Internet]. Vecteezy.com 2020. Available from: <https://www.vecteezy.com/vector-art/292506-a-ph-scale-on-white-background>
- 14 Raspberry Pi Analog to Digital Converters– Adafruit [Internet]. Learn.adafruit.com 2020 [cited 04 May 2020 ]. Available from: <https://learn.adafruit.com/Raspberry-pi-analog-to-digital-converters/mcp3008>
- 15 What is the Raspberry Pi 3? Everything you need to know about the tiny, low-cost computer – CBS Interactive [Internet]. Zdnet.com 2020 [cited ]. Available from: <https://www.zdnet.com/article/what-is-the-Raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/>
- 16 DFRobot/DFRobot\_PH – chocho2018 2018[Internet]. Github.com 2020 [cited 2 November 2018]. Available from: [https://github.com/DFRobot/DFRobot\\_PH/tree/master/RaspberryPi/Python](https://github.com/DFRobot/DFRobot_PH/tree/master/RaspberryPi/Python)
- 17 Integrating Raspberry Pi Sensor Data - via MindConnect IoT Extension – Siemens AG [Internet]. Developer.mindsphere.io 1996 – 2019 [cited 19 December 2018]. Available from: <https://developer.mindsphere.io/howto/howto-rpi-mciot.html>
- 18 This is MindSphere! – Siemens [Internet]. Developer.mindsphere.io 1996 – 2019 [cited 2020]. Available from: <https://new.siemens.com/global/en/products/software/mindsphere.html>
- 19 The psychology behind spending big – BBC [Internet]. BBC.com 2020 [cited 9 October 2017]. Available from: <https://www.bbc.com/worklife/article/20171006-the-psychology-behind-spending-big>
- 20 Cumulocity MQTT Cheat Sheet [Internet]. Cumulocity GmbH; 2017 [cited 2017]. Available from: [http://techcommunity.softwareag.com/documents/portlet\\_file\\_entry/10157/cheatsheet.pdf/31cb975f-20a6-41b5-a932-74e265b22563?status=0](http://techcommunity.softwareag.com/documents/portlet_file_entry/10157/cheatsheet.pdf/31cb975f-20a6-41b5-a932-74e265b22563?status=0)
- 21 Integrating MQTT - via MindConnect IoT Extension – Siemens AG [Internet]. Developer.mindsphere.io 1996 – 2019 [cited 8 October 2018]. Available from: <https://developer.mindsphere.io/howto/howto-mqtt-mciot.html>
- 22 MindAccess - Siemens AG [Internet]. Developer.mindsphere.io 1996 – 2019 [cited 7 February 2020]. Available from: <https://siemens.mindsphere.io/en/docs/mindaccess.html>



- 23 Design of a Opto-Isolator Circuit for Analogue DC Signal [Internet]. S. C. Bera, B. Chakraborty; 2007 [cited September 2007]. Available from: [https://www.sensorsportal.com/HTML/DIGEST/september\\_07/P\\_190.pdf](https://www.sensorsportal.com/HTML/DIGEST/september_07/P_190.pdf)
- 24 Proteus PCB Design and Simulation Software – Introduction - CircuitsToday [Internet]. CircuitsToday.com 2020 [cited 10 November 2019]. Available from: <http://www.circuitstoday.com/proteus-software-introduction>
- 25 Analogue Sensors On The Raspberry Pi Using An MCP3008 - Matt Hawkins [Internet]. Raspberrypi-spy.co.uk 2019 [cited 20 October 2013]. Available from: <https://www.raspberrypi-spy.co.uk/2013/10/analogue-sensors-on-the-raspberry-pi-using-an-mcp3008/>

## Appendix 1. The main Raspberry Pi code

```
Combined_code.py x DFRobot_EC.py x DFRobot_PH.py x ecdata.txt x phdata.txt x
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  import paho.mqtt.client as mqtt
4  import time, threading, ssl, random
5  import RPi.GPIO as GPIO
6  import sys
7  sys.path.append('../')
8
9  import spidev
10 import os
11
12 from DFRobot_EC      import DFRobot_EC
13 from DFRobot_PH     import DFRobot_PH
14
15 # Host/server details and child device name
16 serverUrl  = 'mciotextension.eu1.mindsphere.io'
17 clientId  = 'RaspPi BCM2835 000000008b64f58a'
18 device_name = 'Raspberry Pi 3 Sensor data'
19
20 tenant     = '<TENANT NAME>' #e.g. metropolia
21 username   = '<MCIoT EXTENSION USERNAME>' # e.g. abc@gmail.com
22 password   = '<MCIoT EXTENSION PASSWORD>' #1234567
23
24 receivedMessages = []
25
26 GPIO.setwarnings(False) # Ignore warning for now
27 GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
28 GPIO.setup(29, GPIO.OUT, initial=GPIO.LOW)
29 GPIO.setup(31, GPIO.OUT, initial=GPIO.LOW)
30 # Set pin 29 and 31 to be output pins and set their initial values to low (off)
31
32 ec      = DFRobot_EC()
33 ph      = DFRobot_PH()
34
35 #ph.reset()
36 #ec.reset()
37
38 ec.begin()
39 ph.begin()
40
41 # Open SPI bus
42 spi = spidev.SpiDev()
43 spi.open(0,0)
44 spi.max_speed_hz=1000000
```

```
45
46 # Function to read SPI data from MCP3008 chip
47 # Channel must be an integer 0-7
48 def ReadChannel(channel):
49     adc = spi.xfer2([1, (8+channel)<<4,0])
50     data = ((adc[1]&3) << 8) + adc[2]
51     return data
52
53 # Function to convert data to voltage level,
54 # rounded to specified number of decimal places.
55 def ConvertVolts(data,places):
56     volts = (data * 3.3) / float(1023)
57     volts = round(volts,places)
58     return volts
59
60 # display all incoming messages
61 def on_message(client, userdata, message):
62     print("Received operation " + str(message.payload))
63     if (message.payload.startswith("510")):
64         print("Simulating device restart...")
65         publish("s/us", "501,c8y_Restart");
66         print("...restarting...")
67         time.sleep(1)
68         publish("s/us", "503,c8y_Restart");
69         print("...done...")
70
71 # send measurement
72
73 #For custom measurement use 200,fragment,series,value,unit[,time] in publish
74 #eg 200,myCustomTemperatureMeasurement,fahrenheit,75.2,F
75
76 def sendMeasurements():
77     try:
78         #print("Sending E.C measurement...")
79         publish("s/us", "200,EC,Millisiemens/metres," + str(EC) + ",mS/m")
80
81         #print("Sending pH measurement...")
82         publish("s/us", "200,pH,pH," + str(PH) + ",pH")
83
84         #thread = threading.Timer(7, sendMeasurements)
85         #thread.daemon=True
86         #thread.start()
87         #while True: time.sleep(100)
88     except (KeyboardInterrupt, SystemExit):
```

```

89 |         print("Received keyboard interrupt, quitting ...")
90 |
91 |     # publish a message
92 |     def publish(topic, message, waitForAck = False):
93 |         mid = client.publish(topic, message, 2)[1]
94 |         if (waitForAck):
95 |             while mid not in receivedMessages:
96 |                 time.sleep(0.25)
97 |
98 |     def on_publish(client, userdata, mid):
99 |         receivedMessages.append(mid)
100 |
101 |     def on_connect(client, userdata, flags, rc):
102 |         if rc==0:
103 |             client.connected_flag=True #set flag
104 |             print("connected OK")
105 |         else:
106 |             print("Bad connection Returned code =",rc)
107 |             client.bad_connection_flag=True
108 |
109 |     mqtt.Client.connected_flag=False #create flag in class
110 |     client = mqtt.Client(clientId)
111 |     client.username_pw_set(tenant + "/" + username, password)
112 |     client.on_message = on_message
113 |     client.on_publish = on_publish
114 |
115 |     # connect the client to MindConnect IoT
116 |     client.on_connect=on_connect #bind call back function
117 |     client.loop_start()
118 |     client.connect(serverUrl)
119 |
120 |     #create device (Raspberry Pi 3)
121 |     publish("s/us", "100," + device_name + ",MCIoT_MQTTdevice", True)
122 |     #set required connection interval to 10 minutes
123 |     publish("s/us", "117,10")
124 |
125 |     #create child device
126 |     #publish("s/us", "101,MQTT_Child_device," + device_name + ",MCIoT_MQTTChildType", True)
127 |
128 |     #Send Data via MQTT
129 |     publish("s/us", "110,000000008b64f58a,RaspPi BCM2835,a02082")
130 |     publish("s/us", "114,MCIoT_Restart")
131 |     print("Device registered successfully!")
132 |
133 |     client.subscribe("s/ds")
134 |
135 |     # Define sensor channels
136 |     EC_channel = 0
137 |     PH_channel = 1
138 |
139 |     # Define delay between readings
140 |     delay = 5
141 |
142 |     while True:
143 |
144 |         temperature = 25
145 |
146 |         # Read the EC sensor data
147 |         EC_level = ReadChannel(EC_channel)
148 |
149 |         # Convert EC sensor data to voltage level

```

```
150 EC_volts = ConvertVolts(EC_level,2)
151 EC_millivolts = EC_volts * 1000 #Convert EC value from V to mV
152
153 #Convert voltage to EC with temperature compensation
154 EC_mS_per_centimeter = ec.readEC(EC_millivolts,temperature)
155 EC = EC_mS_per_centimeter * 100
156
157 # Read the PH sensor data
158 PH_level = ReadChannel(PH_channel)
159
160 # Convert PH sensor data to voltage level
161 PH_volts = ConvertVolts(PH_level,2)
162 PH_millivolts = PH_volts * 1000
163
164 #Convert voltage to PH with temperature compensation
165 PH = ph.readPH(PH_millivolts,temperature)
166
167
168 # Print out results
169 print ("-----")
170 print("EC:{}".format(EC))
171 print("pH:{}".format(PH))
172
173
174 sendMeasurements()
175
176
177 if EC > 250 or EC < 150:
178     print("The E.C is beyond acceptable limits!")
179     GPIO.output(29, GPIO.HIGH) # Turn on
180
181 else:
182     GPIO.output(29, GPIO.LOW) # Turn off
183
184 if PH > 6.9 or PH < 6.7:
185     print("The pH is beyond acceptable limits!")
186     GPIO.output(31, GPIO.HIGH) # Turn on led
187
188 else:
189     GPIO.output(31, GPIO.LOW) # Turn off led
190
191 # Wait before repeating loop
192 time.sleep(delay)
```

## Appendix 2. EC and pH meter Raspberry Pi codes

```
DFRobot_EC.py x DFRobot_PH.py x
1  import time
2  import sys
3
4  _kvalue          = 10.0
5  _kvalueLow       = 10.0
6  _kvalueHigh      = 10.0
7  _cmdReceivedBufferIndex = 0
8  _voltage         = 0.0
9  _temperature     = 25.0
10
11 class DFRobot_EC():
12     def begin(self):
13
14         global _kvalueLow
15         global _kvalueHigh
16
17         try:
18             with open('ecdata.txt','r') as f:
19                 kvalueLowLine = f.readline()
20                 kvalueLowLine = kvalueLowLine.strip('kvalueLow=')
21                 _kvalueLow = float(kvalueLowLine)
22                 kvalueHighLine = f.readline()
23                 kvalueHighLine = kvalueHighLine.strip('kvalueHigh=')
24                 _kvalueHigh = float(kvalueHighLine)
25
26         except :
27             print ('ecdata.txt ERROR ! Please run DFRobot_EC_Reset')
28             sys.exit(1)
29
30     def readEC(self,voltage,temperature):
31
32         global _kvalueLow
33         global _kvalueHigh
34         global _kvalue
35
36         rawEC = 100*voltage/820.0/200.0
37         valueTemp = rawEC * _kvalue
38         if(valueTemp > 2.5):
39             _kvalue = _kvalueHigh
40         elif(valueTemp < 2.0):
41             _kvalue = _kvalueLow
42         value = rawEC * _kvalue
43         value = value / (1.0+0.0185*(temperature-25.0))
44         return value
```

```

45
46 def calibration(self,voltage,temperature):
47
48     rawEC = 100*voltage/820.0/200.0
49
50
51     if (rawEC>0.9 and rawEC<1.9):
52
53         compECsolution = 1.413*(1.0+0.0185*(temperature-25.0))
54         KValueTemp = 820.0*200.0*compECsolution/100.0/voltage
55         round(KValueTemp,2)
56         print ('>>>Buffer Solution:1.413mS/cm')
57         f=open('ecdata.txt','r+')
58         flist=f.readlines()
59         flist[0]='kvalueLow='+ str(KValueTemp) + '\n'
60         f=open('ecdata.txt','w+')
61         f.writelines(flist)
62         f.close()
63         print ('>>>EC:1.413mS/cm Calibration completed,Please enter Ctrl+C exit calibration in 5 seconds')
64         time.sleep(5.0)
65
66     elif (rawEC>9 and rawEC<16.8):
67
68         compECsolution = 12.88*(1.0+0.0185*(temperature-25.0))
69         KValueTemp = 820.0*200.0*compECsolution/100.0/voltage
70         print ('>>>Buffer Solution:12.88mS/cm')
71         f=open('ecdata.txt','r+')
72         flist=f.readlines()
73         flist[1]='kvalueHigh='+ str(KValueTemp) + '\n'
74         f=open('ecdata.txt','w+')
75         f.writelines(flist)
76         f.close()
77         print ('>>>EC:12.88mS/cm Calibration completed,Please enter Ctrl+C exit calibration in 5 seconds')
78         time.sleep(5.0)
79
80     else:
81         print ('>>>Buffer Solution Error Try Again<<<')
82
83
84 def reset(self):
85
86     _kvalueLow = 10.0;
87     _kvalueHigh = 10.0;
88

```

```

DFRobot_PH.py
1  import time
2  import sys
3
4  _temperature      = 25.0
5  _acidVoltage      = 2032.44
6  _neutralVoltage   = 1500.0
7  class DFRobot_PH():
8      def begin(self):
9          global _acidVoltage
10         global _neutralVoltage
11         try:
12             with open('phdata.txt','r') as f:
13                 neutralVoltageLine = f.readline()
14                 neutralVoltageLine = neutralVoltageLine.strip('neutralVoltage=')
15                 _neutralVoltage    = float(neutralVoltageLine)
16                 acidVoltageLine    = f.readline()
17                 acidVoltageLine    = acidVoltageLine.strip('acidVoltage=')
18                 _acidVoltage       = float(acidVoltageLine)
19         except :
20             print "phdata.txt ERROR ! Please run DFRobot_PH_Reset"
21             sys.exit(1)
22
23
24     def readPH(self,voltage,temperature):
25         global _acidVoltage
26         global _neutralVoltage
27         slope      = (7.0-4.0)/(((_neutralVoltage-1500.0)/3.0 - (_acidVoltage-1500.0)/3.0)
28         intercept  = 7.0 - slope*((_neutralVoltage-1500.0)/3.0)
29         _phValue   = slope*(voltage-1500.0)/3.0+intercept
30         round(_phValue,2)
31         return _phValue
32
33     def calibration(self,voltage):
34         if (voltage>1322 and voltage<1678):
35             print ">>>Buffer Solution:7.0"
36             f=open('phdata.txt','r+')
37             flist=f.readlines()
38             flist[0]='neutralVoltage='+ str(voltage) + '\n'
39             f=open('phdata.txt','w+')
40             f.writelines(flist)
41             f.close()
42
43             f=open('DFRobot_PH.py','r')
44             list_of_lines = f.readlines()
45             list_of_lines[5]='_neutralVoltage      = '+ str(voltage) + '\n'
46             f=open('DFRobot_PH.py','w')
47             f.writelines(list_of_lines)
48             f.close()
49
50             print ">>>PH:7.0 Calibration completed,Please enter Ctrl+C exit calibration in 5 seconds"
51             time.sleep(5.0)
52

```



```
53 elif (voltage>1854 and voltage<2210):
54     print ">>>Buffer Solution:4.0"
55     f=open('phdata.txt','r+')
56     flist=f.readlines()
57     flist[1]='acidVoltage='+ str(voltage) + '\n'
58     f=open('phdata.txt','w+')
59     f.writelines(flist)
60     f.close()
61
62     f=open('DFRobot_PH.py','r')
63     list_of_lines = f.readlines()
64     list_of_lines[4]='_acidVoltage      = '+ str(voltage) + '\n'
65     f=open('DFRobot_PH.py','w')
66     f.writelines(list_of_lines)
67     f.close()
68
69     print ">>>PH:4.0 Calibration completed,Please enter Ctrl+C exit calibration in 5 seconds"
70     time.sleep(5.0)
71 else:
72     print ">>>Buffer Solution Error Try Again<<<"
73
74 def reset(self):
75     _acidVoltage      = 2032.44
76     _neutralVoltage  = 1500.0
77
78     try:
79         f=open('phdata.txt','r+')
80         flist=f.readlines()
81         flist[0]='neutralVoltage='+ str(_neutralVoltage) + '\n'
82         flist[1]='acidVoltage='+ str(_acidVoltage) + '\n'
83         f=open('phdata.txt','w+')
84         f.writelines(flist)
85         f.close()
86
87         f=open('DFRobot_PH.py','r')
88         list_of_lines = f.readlines()
89         list_of_lines[4]='_acidVoltage      = '+ str(_acidVoltage) + '\n'
90         list_of_lines[5]='_neutralVoltage  = '+ str(_neutralVoltage) + '\n'
91         f=open('DFRobot_PH.py','w')
92         f.writelines(list_of_lines)
93         f.close()
94
95         print ">>>Reset to default parameters<<<"
96     except:
97         f=open('phdata.txt','w')
98         #flist=f.readlines()
99         flist = 'neutralVoltage='+ str(_neutralVoltage) + '\n'
100         flist += 'acidVoltage='+ str(_acidVoltage) + '\n'
101         #f=open('data.txt','w+')
102         f.writelines(flist)
103         f.close()
104         print ">>>Reset to default parameters<<<"
```

## Appendix 3. The EC and pH calibration files

```
DFRobot_EC_Calibration.py x DFRobot_PH_Calibration.py x
1 import sys
2 sys.path.append('../')
3 import spidev
4 import time
5 import os
6 from DFRobot_EC import DFRobot_EC
7 ec = DFRobot_EC()
8 ec.begin()
9
10 # Open SPI bus
11 spi = spidev.SpiDev()
12 spi.open(0,0)
13 spi.max_speed_hz=1000000
14
15 # Function to read SPI data from MCP3008 chip ,Channel must be an integer 0-7
16 def ReadChannel(channel):
17     adc = spi.xfer2([1, (8+channel)<<4,0])
18     data = ((adc[1]&3) << 8) + adc[2]
19     return data
20 # Function to convert data to voltage level,
21 # rounded to specified number of decimal places.
22 def ConvertVolts(data,places):
23     volts = (data * 3.3) / float(1023)
24     volts = round(volts,places)
25     return volts
26
27 # Define sensor channels
28 EC_channel = 0
29 while True :
30     #Read your temperature sensor to execute temperature compensation
31     temperature = 25
32
33     # Read the EC sensor data
34     EC_level = ReadChannel(EC_channel)
35
36     # Convert EC sensor data to voltage level
37     EC_volts = ConvertVolts(EC_level,2)
38     EC_millivolts = EC_volts * 1000 #Convert EC value from V to mV
39     print("CH0: {}mV".format(EC_millivolts))
40
41     #Calibrate the calibration data
42     ec.calibration(EC_millivolts,temperature)
43     time.sleep(3.0)
```

```
DFRobot_EC_Calibration.py x DFRobot_PH_Calibration.py x
1 import sys
2 sys.path.append('../')
3 import spidev
4 import time
5 import os
6 from DFRobot_PH import DFRobot_PH
7 ph = DFRobot_PH()
8 ph.begin()
9
10 # Open SPI bus
11 spi = spidev.SpiDev()
12 spi.open(0,0)
13 spi.max_speed_hz=1000000
14
15 # Function to read SPI data from MCP3008 chip, Channel must be an integer 0-7
16 def ReadChannel(channel):
17     adc = spi.xfer2([1,(8+channel)<<4,0])
18     data = ((adc[1]&3) << 8) + adc[2]
19     return data
20 # Function to convert data to voltage level,
21 # rounded to specified number of decimal places.
22 def ConvertVolts(data,places):
23     volts = (data * 3.3) / float(1023)
24     volts = round(volts,places)
25     return volts
26
27 # Define sensor channels
28 PH_channel = 1
29 while True :
30     #Read your temperature sensor to execute temperature compensation
31     temperature = 25
32
33     # Read the pH sensor data
34     PH_level = ReadChannel(PH_channel)
35
36     # Convert pH sensor data to voltage level
37     PH_volts = ConvertVolts(PH_level,2)
38     PH_millivolts = PH_volts * 1000 #Convert pH value from V to mV
39     print("CH1: {}mV".format(PH_millivolts))
40
41     #Calibrate the calibration data
42     ph.calibration(PH_millivolts,temperature)
43     time.sleep(3.0)
44
```

## Appendix 4. How to connect a device to MindSphere(MindConnect)

### 1. Sign in to MindSphere

# SIEMENS

## Sign In

or [create an account](#)

---

**Sign In has changed.** If you previously signed in with a username, please use your email. [Need help?](#)

Email

Password [Show](#)

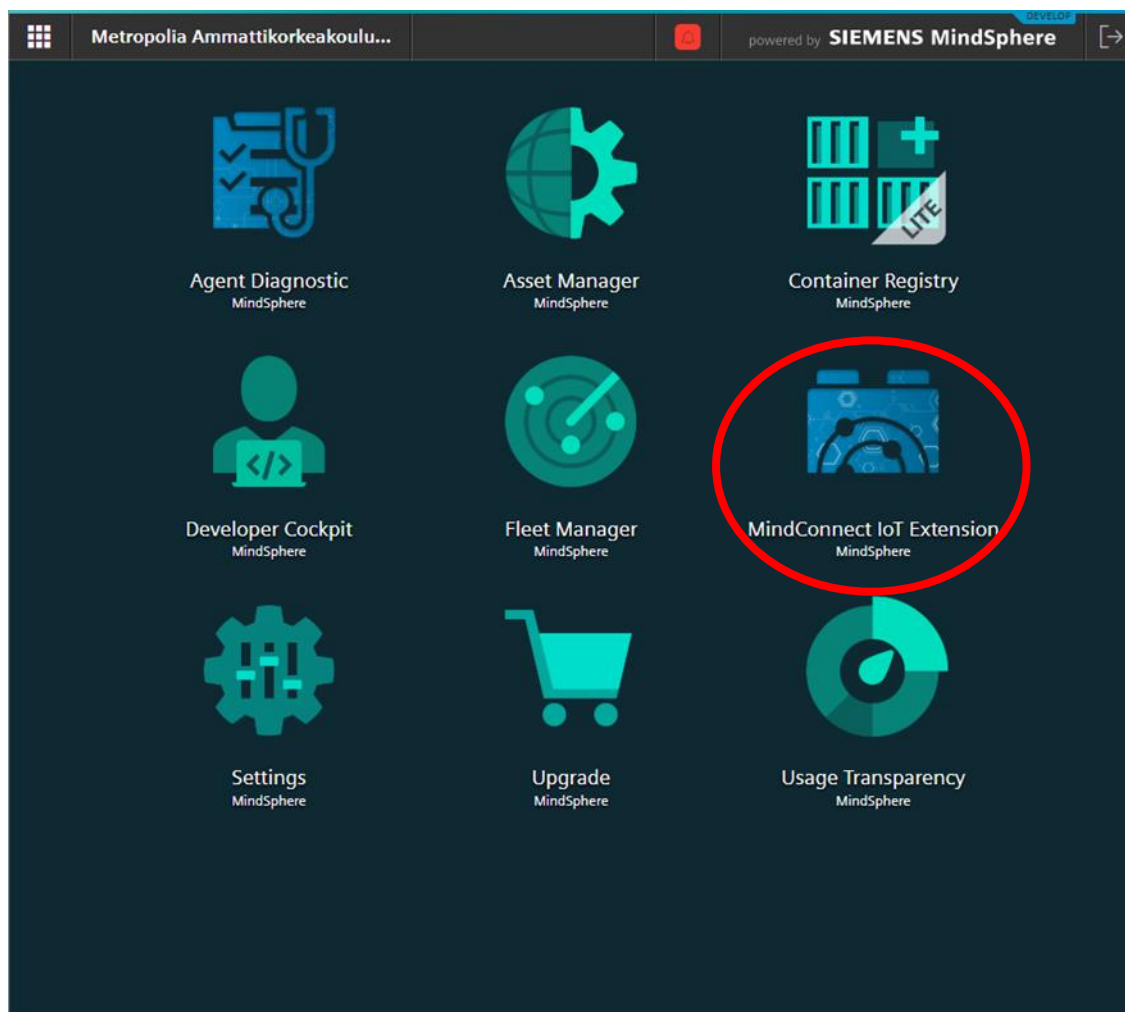
[Forgot your password?](#)

**Sign In**

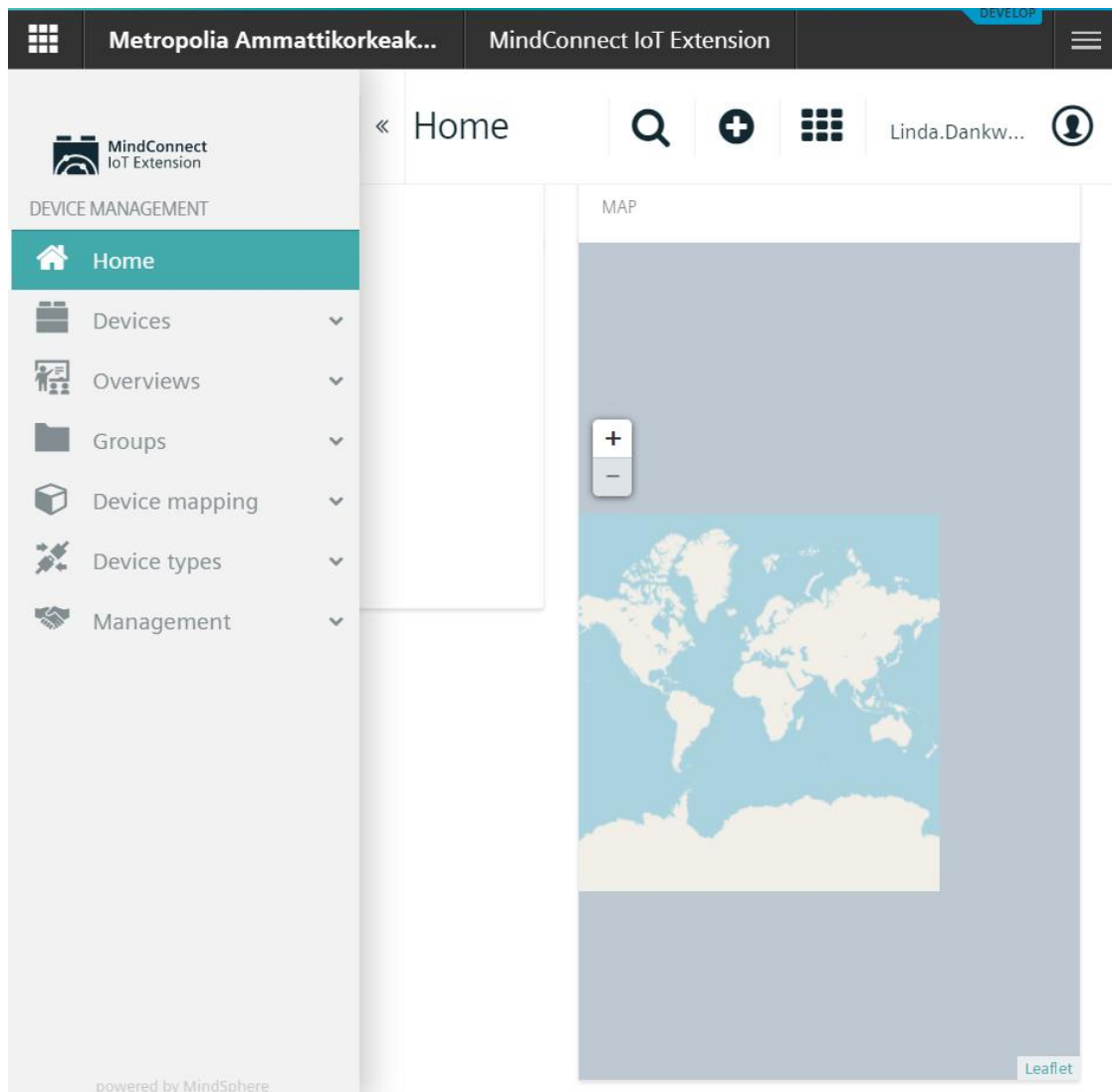
©2020 Siemens Digital Industries Software

[Privacy](#) [Terms](#) [Help](#)

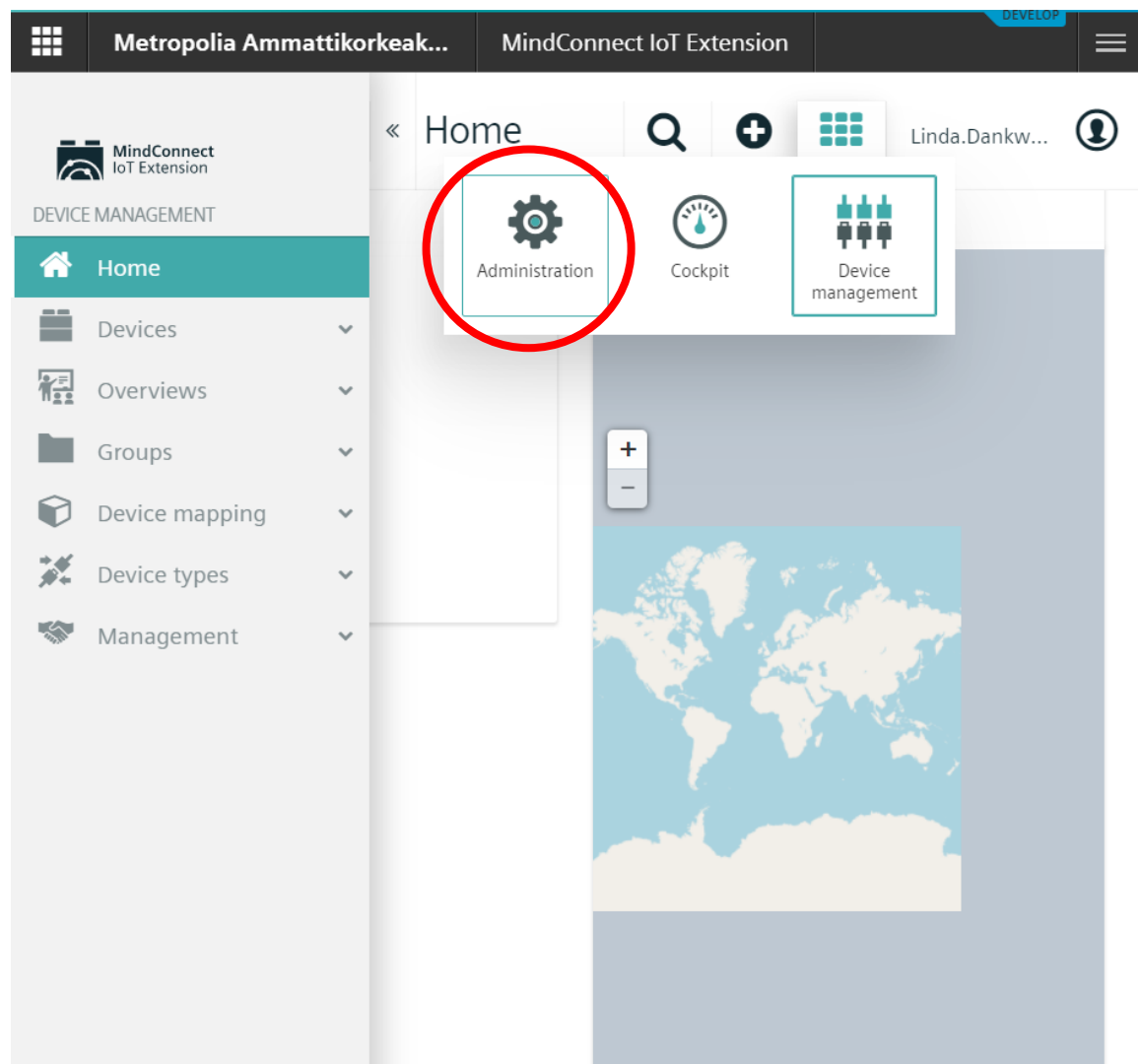
2. Go to the Launchpad, and click on the MindConnect IoT extension.



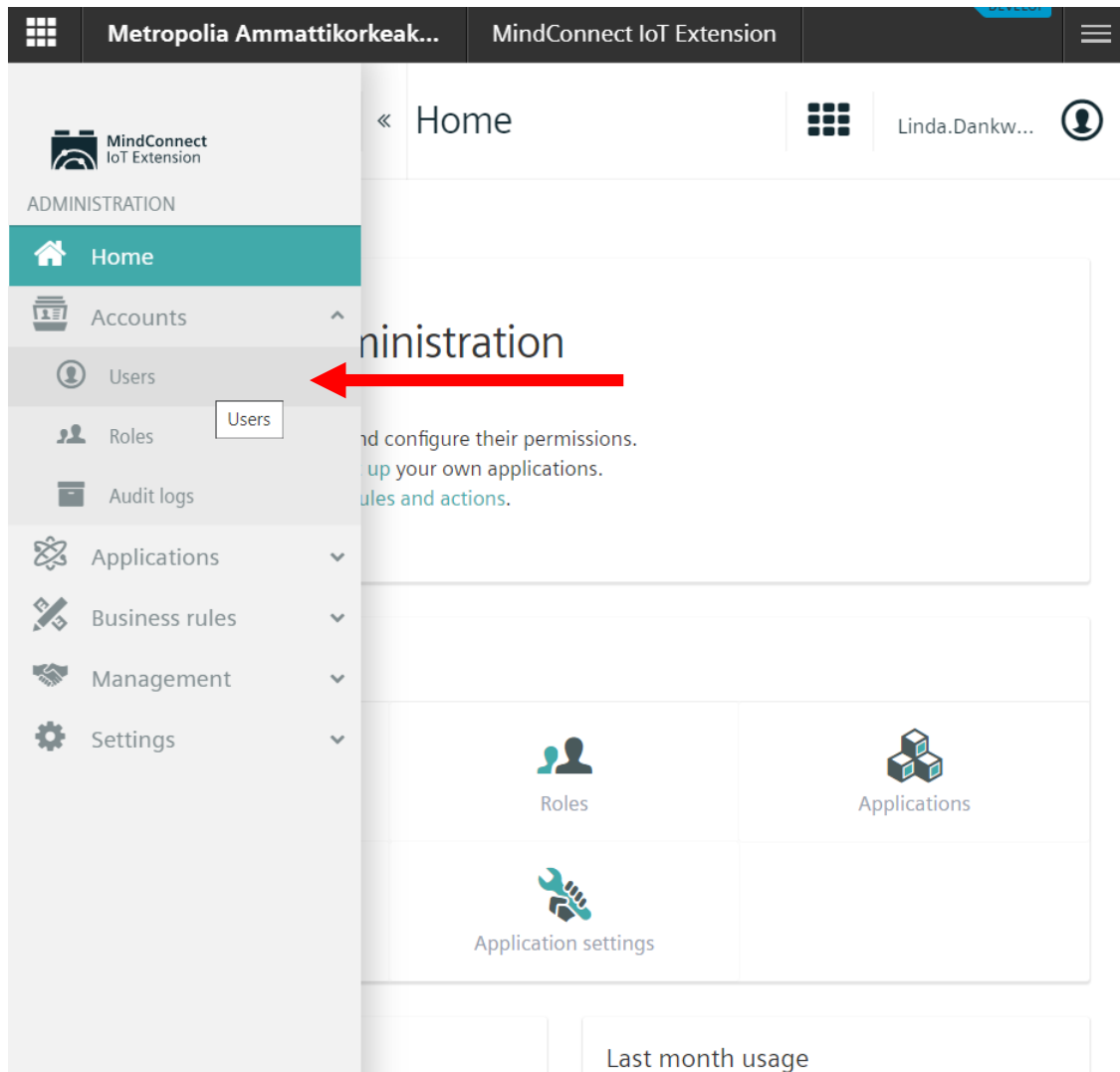
3. The link will usually redirect the user to the device management section of MindConnect



## 4. Switch over to the administration section of the MindConnect extension



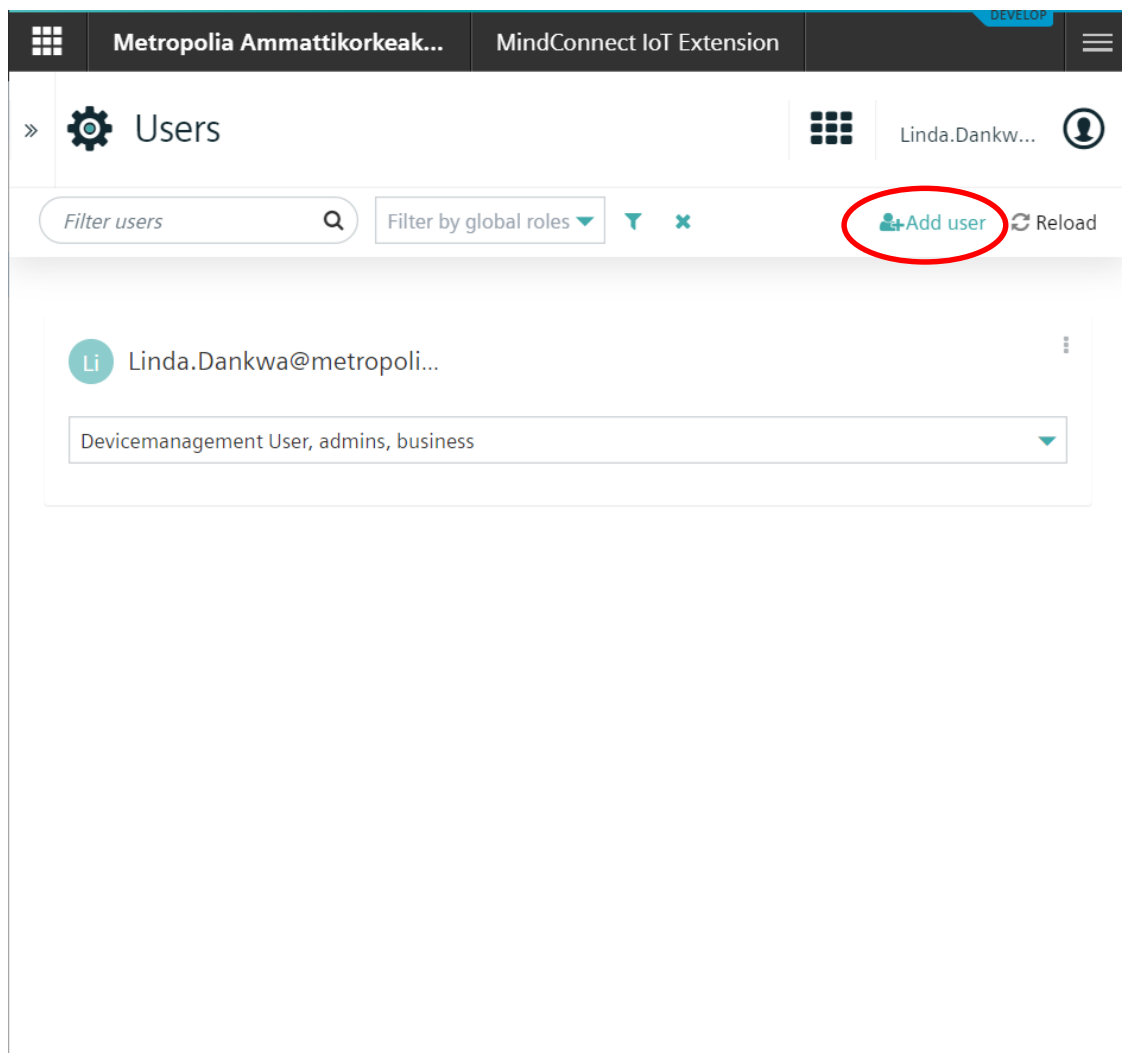
5. Open the Accounts tab on the navigator and click on Users



The screenshot displays the MindConnect IoT Extension administration interface. The top navigation bar includes the Metropolia logo, the text "Metropolia Ammattikorkeak...", "MindConnect IoT Extension", and a user profile icon for "Linda.Dankw...". The left sidebar, titled "ADMINISTRATION", contains the following menu items: Home (selected), Accounts, Users (highlighted with a red arrow), Roles, Audit logs, Applications, Business rules, Management, and Settings. The main content area shows the "Home" page with a "Users" tile and a "Roles" tile. The "Users" tile is highlighted with a red arrow. The "Roles" tile is also visible. The "Applications" tile is visible. The "Application settings" tile is visible. The "Last month usage" tile is visible.



## 6. Click on the Add user button



The screenshot shows the Metropolia Users management interface. The top navigation bar includes the Metropolia logo, the text "Metropolia Ammattikorkeak...", "MindConnect IoT Extension", and a "DEVELOP" badge. The main header area displays "Users" with a gear icon, a user profile for "Linda.Dankw...", and a search bar labeled "Filter users". Below the search bar, there is a "Filter by global roles" dropdown menu and a "Reload" button. The "Add user" button, which features a person icon and a plus sign, is circled in red. Below the navigation bar, a user card is visible for "Linda.Dankwa@metropoli...", showing a profile picture with the initials "Li" and a dropdown menu for roles containing "Devicemanagement User, admins, business".

7. Add a new user to the MindConnect system by filling in all the required information. This new user has to be completely unassociated with the other MindSphere applications, otherwise the connection to the MindConnect Server will fail. After their account is created, they must only be able to login to the MindConnect IoT extension directly and must not be able to login through the Launchpad.

The screenshot shows the 'New user' form in the MindConnect IoT Extension. The form is titled 'New user' and includes a 'Users' icon. The form is divided into several sections: 'Identification', 'Login alias', 'Email', 'First name', 'Last name', 'Telephone', and 'Login options'. The 'Identification' section contains a 'Username (e.g. email)' field with a placeholder 'e.g. joe.doe@example.com (required)', a 'Status' toggle set to 'Enabled', and a 'Login alias' field with a placeholder 'e.g. joe.doe'. The 'Email' section contains an 'Email' field with a placeholder 'e.g. joe.doe@example.com (required)' and a red warning message: 'Beware that email is required to reset password.' The 'First name' and 'Last name' sections each have an empty text input field. The 'Telephone' section has a text input field with a placeholder 'e.g. +49 9 876 543 210'. The 'Login options' section contains three checkboxes: 'Two-factor authentication (SMS)' (unchecked), 'User must reset password on next login' (unchecked), and 'Send password reset link as email' (checked).

Metropolia Ammattikorkeak... MindConnect IoT Extension

» New user Linda.Dankw...

Users

Identification

Username (e.g. email) Status

Enabled

Login alias

Email

Beware that email is required to reset password.

First name Last name

Telephone

Login options

Two-factor authentication (SMS)

User must reset password on next login

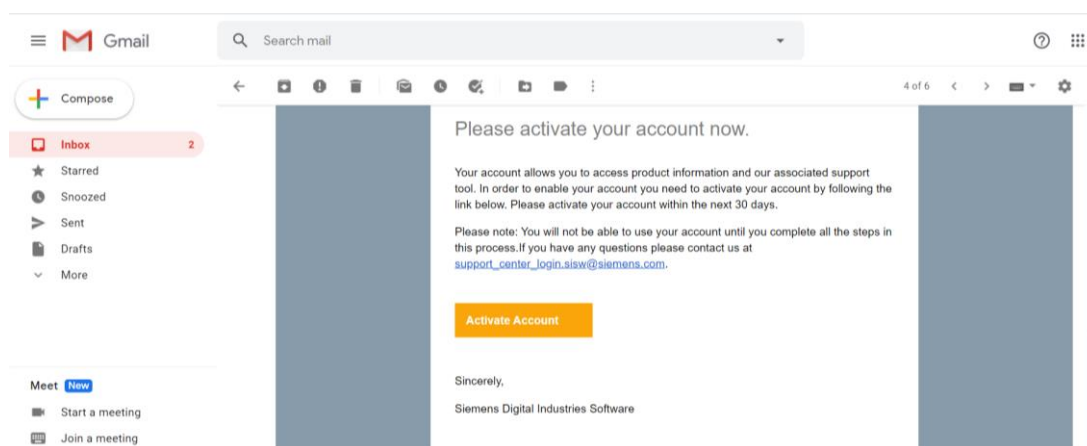
Send password reset link as email

8. Scroll to the Global roles subheading and give the new user Devicemanagement User and admins roles, by clicking the checkboxes. Then click save

The screenshot shows the 'New user' configuration interface in MindSphere. The 'Global roles' section is expanded, displaying a list of roles with checkboxes. The 'admins' and 'Devicemanagement User' roles are checked. The 'Save' button is highlighted.

Role	Description	Selected
admins		<input checked="" type="checkbox"/>
business		<input type="checkbox"/>
CEP Manager	Has full access to all deployed CEP modules and SmartRules	<input type="checkbox"/>
Cockpit User	User to work in Cockpit application. This does not include the access to any device data.	<input type="checkbox"/>
Devicemanagement User	Gives access to bulk operations and device management application. This does not include access to any device data.	<input checked="" type="checkbox"/>
devices		<input type="checkbox"/>
Global Manager	Can read and write all data from all devices	<input type="checkbox"/>
Global Reader	Can read all data from all devices	<input type="checkbox"/>
Global User Manager	Can access and edit the full user hierarchy	<input type="checkbox"/>
readers		<input type="checkbox"/>
Shared User Manager	Can create new user as his own sub-users and manage this	<input type="checkbox"/>
Tenant Manager	Can manage tenant wide configurations like applications, tenant options and retention rules	<input type="checkbox"/>

9. At this point, the new user will get an email from MindSphere to activate their account and reset their password.

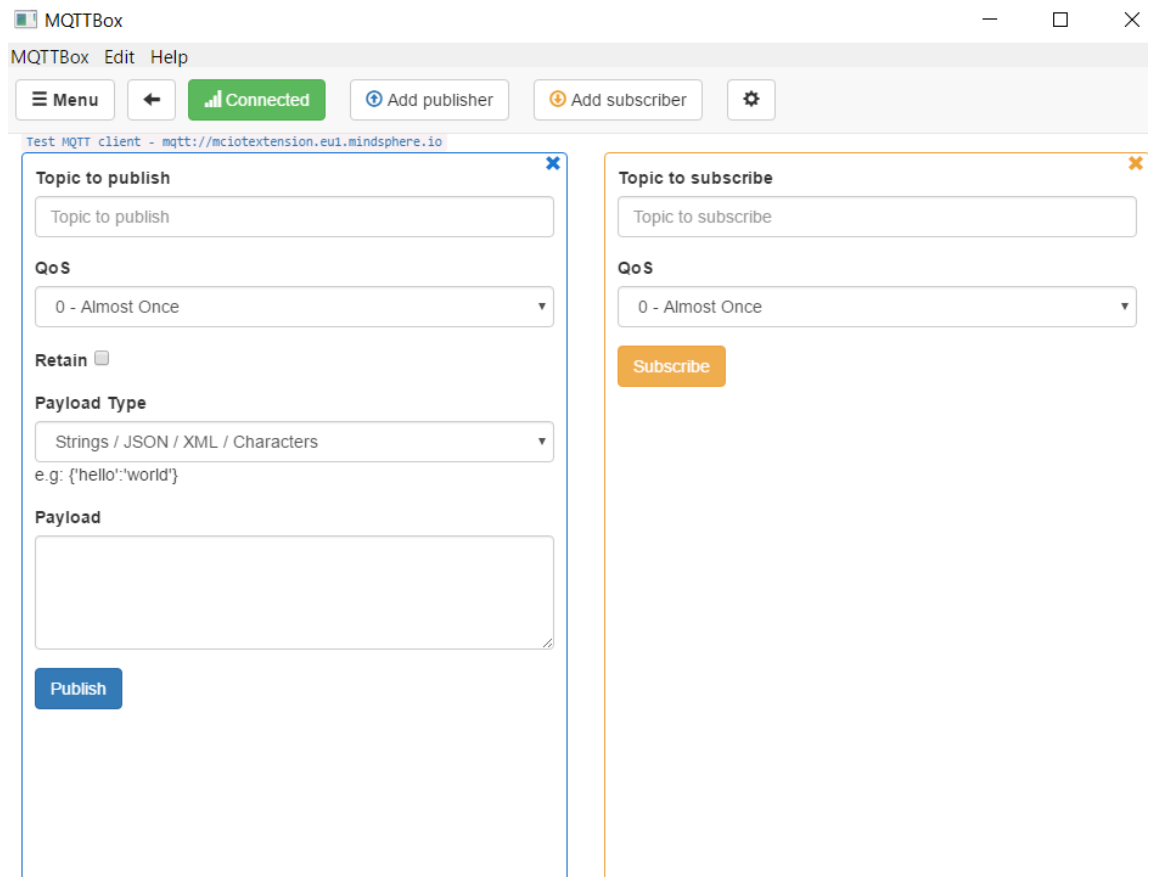


This will be the account used every time a connection to MindConnect is made from an outside source. Connection to the server itself (by MQTT) can be done in one of two ways, firstly by using an MQTT client application on a PC like MQTTBox or by using written down code (in any programming language).

10. To connect via MQTTBox, open MQTTBox and click Create MQTT Client. Fill in the information listed below into the form for "MQTT CLIENT SETTINGS", then click save. Fields that are not listed can be left as default.

Field	Configuration
MQTT Client Name	<code>{uniqueName}</code>
MQTT Client ID	<code>{arbitraryUniqueID}</code>
Protocol	<code>mqtt/tcp</code>
Host	<code>mciotextension.{YourRegion}.{mindsphere-domain}</code> e.g.: <code>mciotextension.eu1.mindsphere.io</code>
Username	<code>{tenantName}/{mciotExtensionUsername}</code>
Password	<code>{password}</code>
Append timestamp to MQTT client ID?	unchecked
Broker is MQTT v3.1.1 compliant	checked
Clean Session	checked
Queue outgoing QoS zero messages	checked
Keep alive	checked

11. A connection to the MindConnect broker will then be established



12. To connect to MindConnect via some type of programming language, very specific code has to be written. The code shown below can be run on all devices that support python.

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  import paho.mqtt.client as mqtt
4  import time, threading, ssl, random
5
6  # Host/server details and child device name
7  serverUrl = 'mciotextension.eul.mindsphere.io'
8  clientId = 'RaspPi BCM2835 000000008b64f58a'
9  device_name = 'Raspberry Pi 3 Sensor data'
10
11  tenant = 'TENANT'
12  username = 'EMAIL ADDRESS'
13  password = 'PASSWORD'
14
15  receivedMessages = []
16
17  # display all incoming messages
18  def on_message(client, userdata, message):
19      print("Received operation " + str(message.payload))
20      if (message.payload.startswith("510")):
21          print("Simulating device restart...")
22          publish("s/us", "501,c8y_Restart");
23          print("...restarting...")
24          time.sleep(1)
25          publish("s/us", "503,c8y_Restart");
26          print("...done...")
27
28  # send measurement
29
30  #For custom measurement use 200,fragment,series,value,unit[,time] in publish
31  #eg 200,myCustomTemperatureMeasurement,fahrenheit,75.2,F
32
33  def sendMeasurements():
34      try:
35          print("Sending E.C measurement...")
36          EC = str(random.randint(0, 2000))
37          publish("s/us", "200,EC,Millisiemens/metres," + str(EC) + ",mS/m")
38
39          print("Sending pH measurement...")
40          PH = str(random.randint(0, 14))
41          publish("s/us", "200,pH,pH," + str(PH) + ",pH")
42
43          thread = threading.Timer(7, sendMeasurements)

```

```

44     thread.daemon=True
45     thread.start()
46     while True: time.sleep(100)
47     except (KeyboardInterrupt, SystemExit):
48         print("Received keyboard interrupt, quitting ...")
49
50     # publish a message
51     def publish(topic, message, waitForAck = False):
52         mid = client.publish(topic, message, 2)[1]
53         if (waitForAck):
54             while mid not in receivedMessages:
55                 time.sleep(0.25)
56
57     def on_publish(client, userdata, mid):
58         receivedMessages.append(mid)
59
60     def on_connect(client, userdata, flags, rc):
61         if rc==0:
62             client.connected_flag=True #set flag
63             print("connected OK")
64         else:
65             print("Bad connection Returned code =",rc)
66             client.bad_connection_flag=True
67
68     mqtt.Client.connected_flag=False #create flag in class
69     client = mqtt.Client(clientId)
70     client.username_pw_set(tenant + "/" + username, password)
71     client.on_message = on_message
72     client.on_publish = on_publish
73
74     # connect the client to MindConnect IoT
75     client.on_connect=on_connect #bind call back function
76     client.loop_start()
77     client.connect(serverUrl)
78
79     #create device (Raspberry Pi 3)
80     publish("s/us", "100," + device_name + ",MCIoT_MQTTdevice", True)
81     #set required connection interval to 10 minutes
82     publish("s/us", "117,10")
83
84     #create child device
85     #publish("s/us", "101,MQTT_Child_device," + device_name + ",MCIoT_MQTTChildType", True)
86
87     #Send Data via MQTT
88     publish("s/us", "110,000000008b64f58a,RaspPi BCM2835,a02082")
89     publish("s/us", "114,MCIoT_Restart")
90     print("Device registered successfully!")
91
92     client.subscribe("s/ds")
93
94     #send random numbers as a test
95     EC = str(random.randint(0, 2000))
96     PH = str(random.randint(0, 14))
97
98     sendMeasurements()
99

```

13. To connect a Raspberry Pi to the IoT extension, a few steps must occur before running the code above. Firstly, the python packages in the Pi device need to be updated. This is done by running the code below on the terminal of the Raspberry Pi.

```
sudo apt-get update
```

14. The python library paho-mqtt also needs to be installed by running the commands below on the terminal.

```
sudo apt-get install python3-pip
```

```
sudo pip3 install paho-mqtt
```