Hatim Mala

Hatim.mala@han.nl

A step-by step procedure to set up the hardware, firmware, and controller software of the Open-Source Energy Manager

Getting Started With OSEM

A user’s Guide

Contents

[1. About this document 2](#_Toc121836631)

[2. About OSEM 2](#_Toc121836632)

[3. Hardware Requirements 4](#_Toc121836633)

[3.1 The controller hardware: 4](#_Toc121836634)

[3.2 Sensors: 5](#_Toc121836635)

[3.2.1 Temperature Sensors 5](#_Toc121836636)

[3.2.2 Flow sensor 6](#_Toc121836637)

[4. Software Requirements: 8](#_Toc121836638)

[4.1 MATLAB/Simulink: 8](#_Toc121836639)

[4.2 Mu Editor 8](#_Toc121836640)

[4.3 System Workbench for STM32 8](#_Toc121836641)

[5. Setting up the system: 9](#_Toc121836642)

[5.1 Programming the PICO microcontroller 9](#_Toc121836643)

[5.2 Setting up the raspberry PI 11](#_Toc121836644)

[5.2.1 Simulink Support Package for Raspberry PI 11](#_Toc121836645)

[5.2.2 Configuring the Raspberry PI Operating System 12](#_Toc121836646)

[6. Deploying the MPC controller into the Raspberry PI Hardware 13](#_Toc121836647)

[6.1 The Simulink File: 13](#_Toc121836648)

[6.2 Deployment Procedure: 15](#_Toc121836649)

# About this document

This document is intended as a guide for users, developers or researchers who wish to experiment with or use the Open-source Energy Manager (OSEM). This document covers the following aspects:

* Briefly explain what OSEM is and the main system architecture.
* List the required Hardware of the system.
* List the required software.
* Procedure for downloading the various firmware to the microprocessors.
* Verifying and debugging the hardware connections.

Note that the scope of his document is only restricted to the procedure of setting up and starting the system. The theoretical aspects of designing the control algorithm is available on a separate documentation in [OSEM GitHub repository](https://github.com/hancse/OSEM) .

# About OSEM

OSEM (Open-Source Energy Manager) is a model-predictive controller (MPC) for a plant consisting of a heat pump and a stratified water storage tank. The control strategy follows the standard MPC presentation depicted in the figure below.

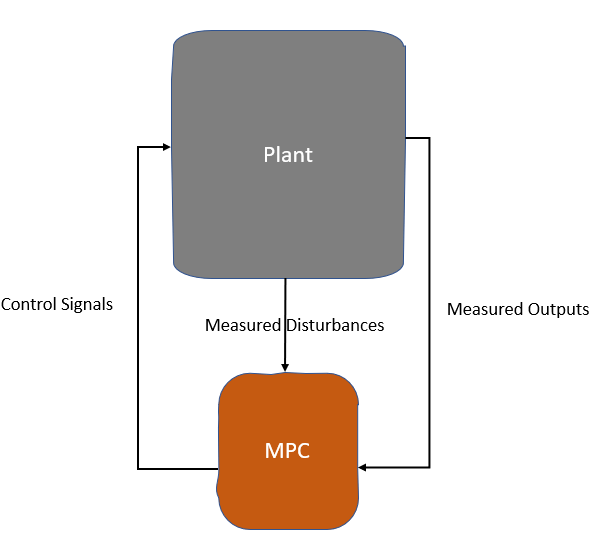


Figure 1: Standard MPC Architecture

When this standard architecture is applied to this application, it results in the control structure depicted in the figure below. In work package 4 documentation, a detailed analysis and modelling of the plant, along with the theoretical justification of the choices, the design procedure of the MPC controller are reported.

**The plant** consists of an air/water **heat pump**. This heat pump is controlled with a signal between 0-100% representing the power of the heat pump compressor. The plant also contains a stratified **water storage tank**. The tank is connected to the heat pump via a spiral heat exchanger inside the tank. The tank also contains an **electric heating element**. The heating element is controlled by a binary signal corresponding to ON/OFF state.

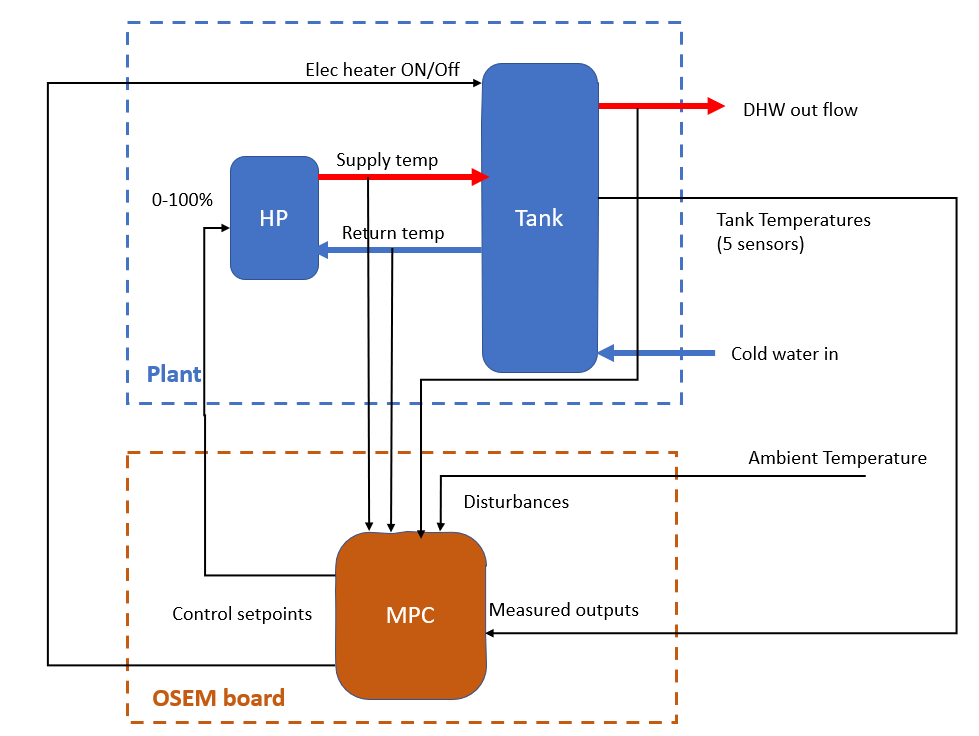


Figure 2: Overview of the control structure

The tank is fitted with 5 temperature sensors fitted laterally across the tank height. Those five temperatures represent the plant **measured outputs** and are fed back to the MPC controller. In addition, the plant contains a flow sensor to measure the DHW tapping pattern, which is fed back to the controller as a disturbance. Moreover, the supply and return temperatures of the heat pump are also measured and feedback as **disturbances** to the controller.

# Hardware Requirements

## The controller hardware:

The controller hardware is shown in the figure below. In this document, the main components, architecture, and functionality will be explained. For the detailed circuit diagram of this board, please refer to Contecto’s documentation on GitHub.

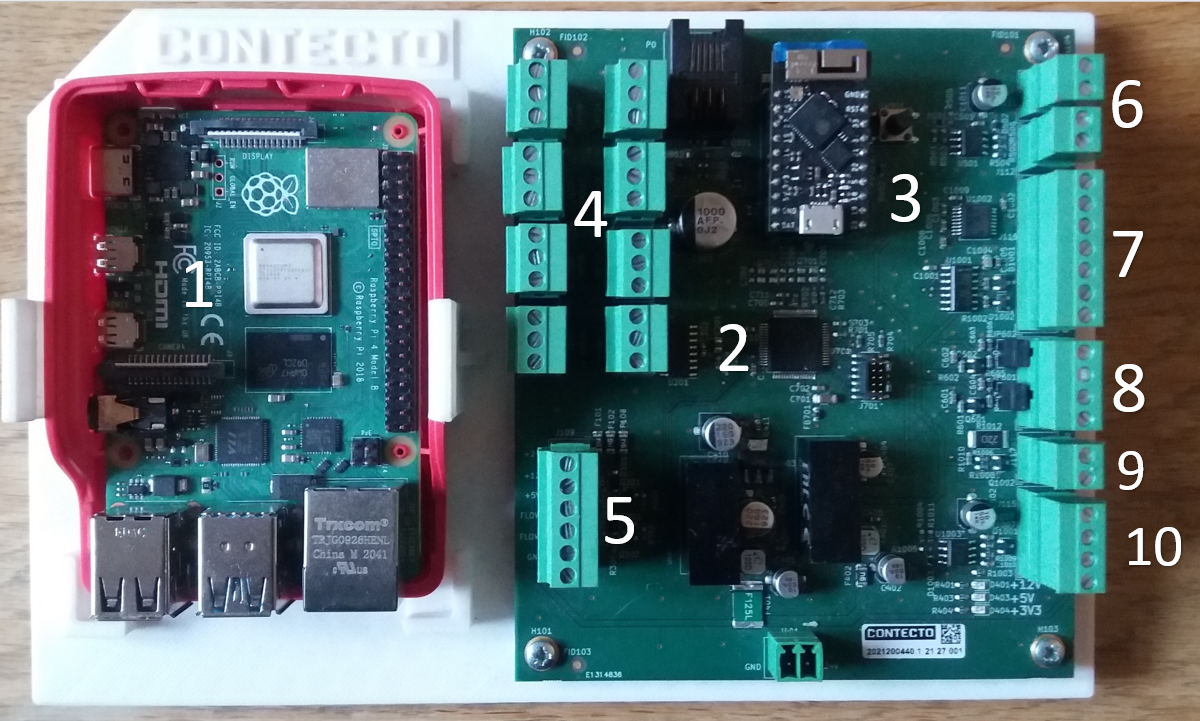


Figure 3: Overview of the controller Hardware board

1. **Raspberry PI 4 Model B:** On this computer, the model-predictive controller code will be deployed.
2. **STM 32 Microcontroller:** This microcontroller is responsible for acquiring data from the sensors and sending setpoints to the actuators.
3. **PICO Microcontroller:** Acquires the sensor data from and sends setpoints to the STM32 via I2C communication. The PICO also forms a WIFI network and communicates with the PI via MQTT protocol over WIFI.
4. Input ports for connecting the temperature sensors (8 sensors).
5. Input ports for connecting flow sensors (2 Sensors).
6. Output ports for valve control (2 valves).
7. UART Communication port.
8. I2C communication port.
9. 5V digital output port.
10. 24V output port.

**Important Note:**

The OSEM board shown above was originally designed to control a system for space heating and DHW water. However, since the 1st iteration of the project was restricted to DHW control only, the OSEM board contains functionalities that are not utilized in this application but are available to accommodate future developments. For example, in the DHW control application; no valve control is used. Also, only one flow sensor is used.

The three microprocessors work together as follows: The STM 32 microcontroller is connected directly to the sensors (Temperature sensors and flow sensors). The STM 32 acquires the data from the sensors and sends it over I2C the PICO microprocessor. The PICO microcontroller is equipped with a WIFI module to which the Raspberry PI also connects. The sensor data is then passed from the PICO to the PI using MQTT protocol. ON the PI, the MPC code collects the sensor data and decides the control action, this control action is sent to the PICO via MQTT protocol over WIFI. The PICO then passes this control setpoint to the STM 32 via I2C. The STM 32 then actuates the heat pump via the outport shown in figure 1. This architecture is depicted in figure 2 below:

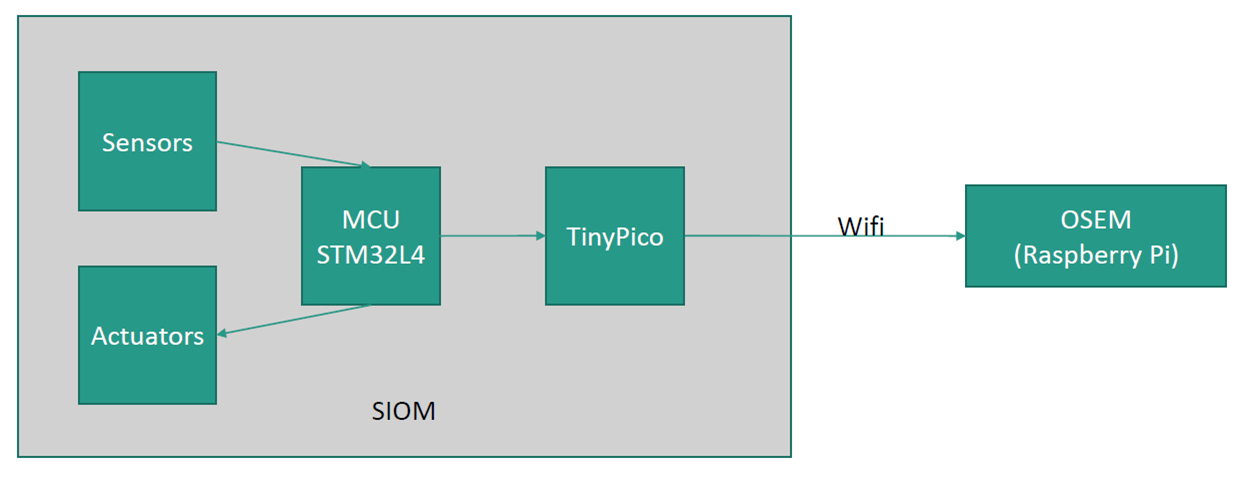


Figure 4: Overview of the hardware and communication architecture

## Sensors:

### Temperature Sensors

The temperature sensor used in this application is DS18B20. The number of sensors required is 8, arranged as follows: 5 sensors to measure the temperature across the storage tank, 2 sensors to measure the feed and return of the heat pump, and 1 sensor for ambient temperature. Figure 4 shows the locations where the sensors are installed. It is important to obey this numbering shown in the figure, because the controller and communication software addresses the sensors by those numbers.



Figure 5: temperature sensor DS18B20

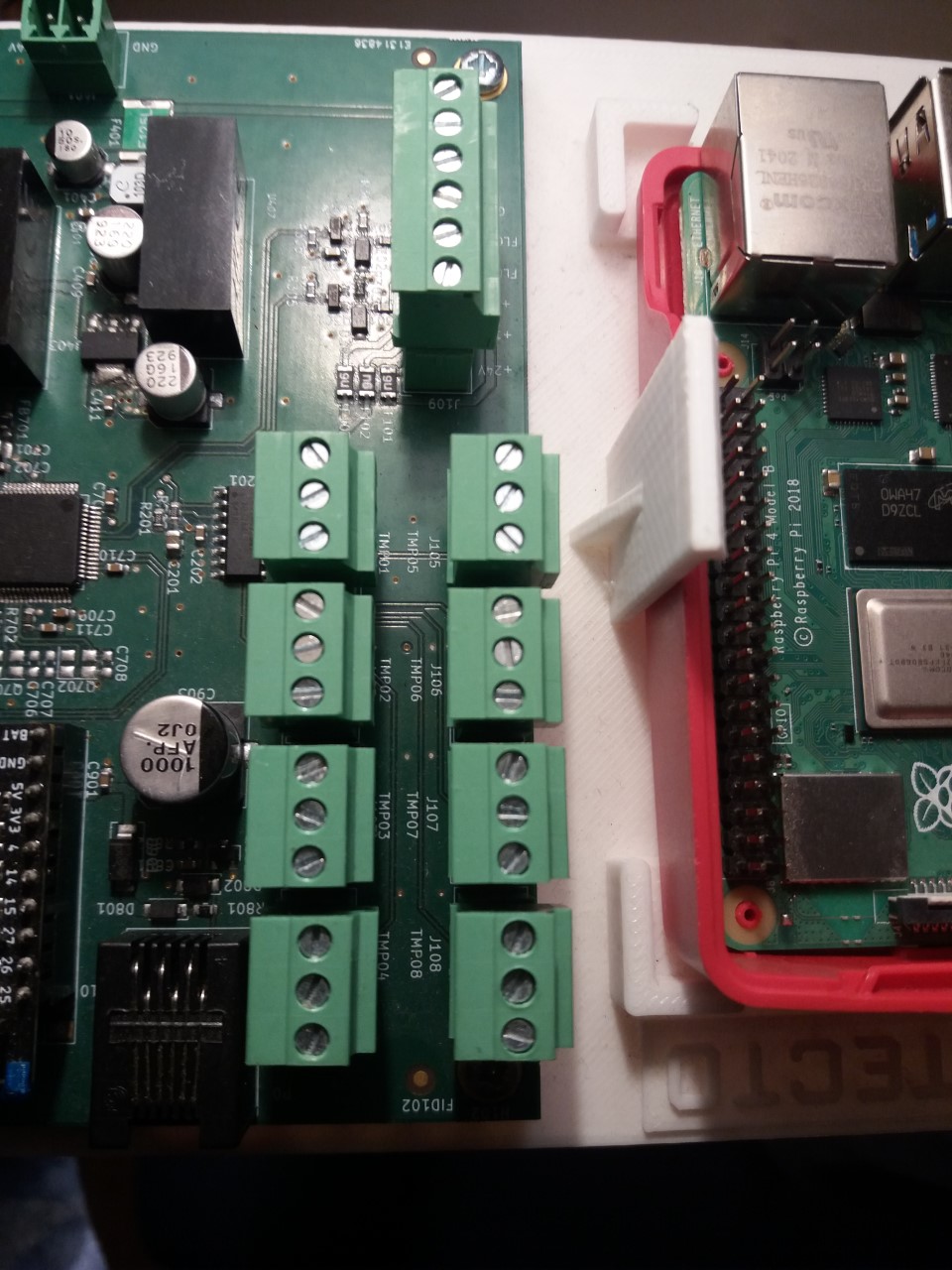


Figure 6: Ports on OSEM board to connect the 8 temperature sensors

### Flow sensor

The flow sensor in this application is Water Flow Sensor YF-B6. This sensor measures the DHW flow (Hot water tapping) and is placed as shown in figure 8.

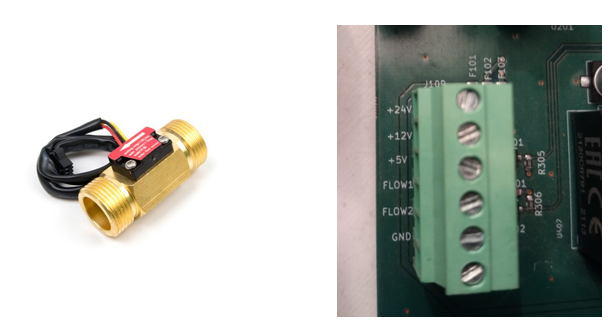


Figure 7: YF-B6 flow sensor (Left) and the port for connecting the sensor to OSEM board

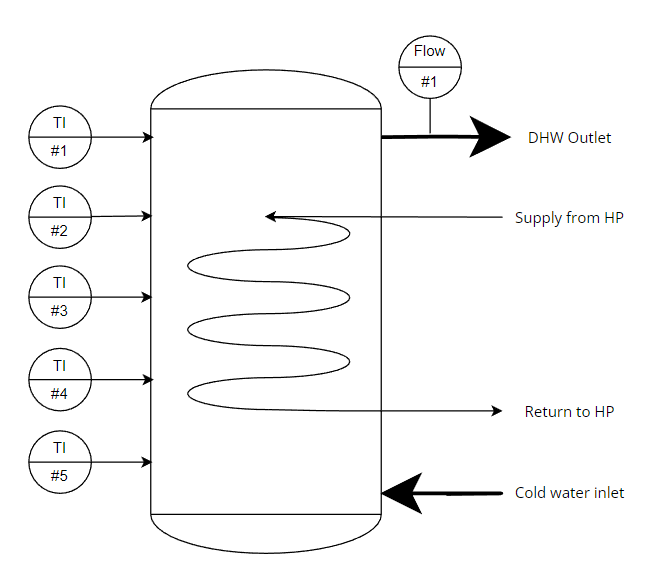


Figure 8: Location of the sensors in the plant

# Software Requirements:

## MATLAB/Simulink:

In this application, MATLAB/Simulink is used to develop the dynamic model of the plant, design the MPC controller and simulate the controlled system. It is also used to generate the MPC code that runs on the Raspberry PI. MATLAB/Simulink 2020b (or a more recent version) is required. The following toolboxes need to be installed:

* Data Acquisition toolbox.
* Signal processing Toolbox.
* DSP System Toolbox.
* Model Predictive Control toolbox.
* Control System toolbox.
* Embedded Coder.
* MATLAB Coder.
* Simulink Coder.

## Mu Editor

Mu is a simple-to-use, free and open-source Python editor and IDE (integrated development environment). In this application, Mu is used to program the PICO microcontroller. Mu can be downloaded from [here](https://codewith.mu/en/download).

## System Workbench for STM32

System Workbench for STM32 is a free IDE toolchain, which can be downloaded from [here](https://www.st.com/en/development-tools/sw4stm32.html). In addition, STM32CubeMX software is required, which can be downloaded from this [link](https://www.st.com/en/development-tools/stm32cubemx.html). STM32CubeMX is a graphical tool that allows a very easy configuration of STM32 microcontrollers and microprocessors, as well as the generation of the corresponding initialization C code.

# Setting up the system:

## Programming the STM32 Microcontroller

Note: This section assumes some familiarity with STM32 Microcontrollers flashing. To familiarize yourself with the topic, you can watch this [video](https://www.youtube.com/watch?v=dnfuNT1dPiM&ab_channel=RobertFeranec) introduction. In addition to the tutorials on STM32electronics YouTube page.

Download the files from the OSEM GitHub repository

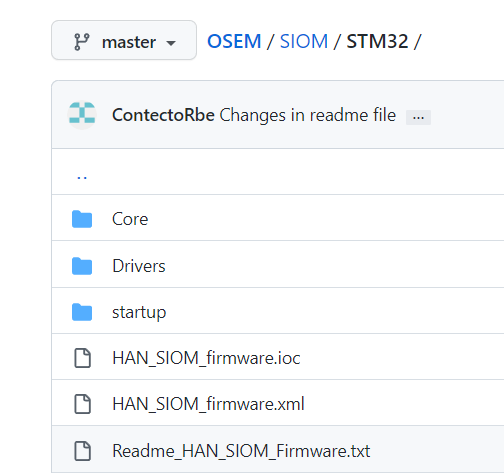


Figure 9: Directory in the OSEM Repository for STM code

In order to run the project import project via "File >> Import >> general >> Existing projects into workspace" and select the “.cproject” file. When settings are changed in cubeMx, generate the file with basic application structure + SW4STM32 Toolchain.

By default, the STM controller program connector is also the debug UART output connector, this is particularly handy when using the STLink V3 programmer as 1 connector is used for programming and debug UART. The connector used is the standard STLink V3 connector which comes with the STLink V3.

The system Debug output can be changed by editing the "systemInit" function in the main.c file >> the given UART handle refers to the debug output

Should any error occur, this will be outputted via the debug UART, the default settings are:

- Baudrate = 115200

- Data bibts: 8

- parity bits: None

- Stop bits: 1

This can be opened by any standard communication terminal program (Putty, tera term, Realterm etc.). System important settings (such as definitons and addresses)and structure are defined in the 'SystemDefinitons.h/.c' files.

## Programming the PICO microcontroller

The PICO microcontroller acts as the “middleman” between the STM32 Microcontroller and the Raspberry PI. The PICO microcontroller receives the sensor readings from the STM32 via I2C, packages the sensor readings into MQTT messages, forms a WIFI network, and sends the MQTT messages over this WIFI to the raspberry PI. In this section, the procedure for downloading the python codes on the PICO microcontroller to execute these tasks is explained.

First, download the following Python files from the OSEM GitHub Repository:

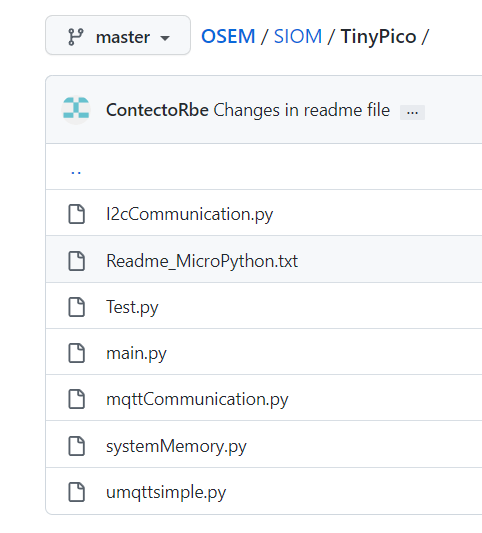


Figure 10: Python files for the PICO microcontroller

Next, Open Mu editor. On the main panel, click “Load”.

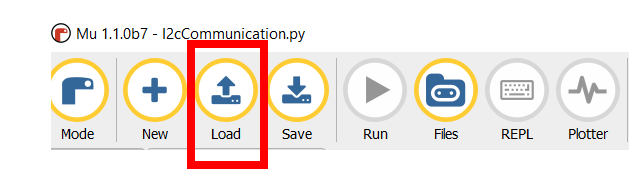


Figure 11: Loading the Python files into Mu editor

Navigate to the directory where you downloaded the files in the previous step and load the files.

Connect the PICO microcontroller to your PC via USB and click on “Files” on Mu editor main panel. This will show a menu with a list of the files on your computer, and a list of the files currently in the PICO. As shown in the next figure.

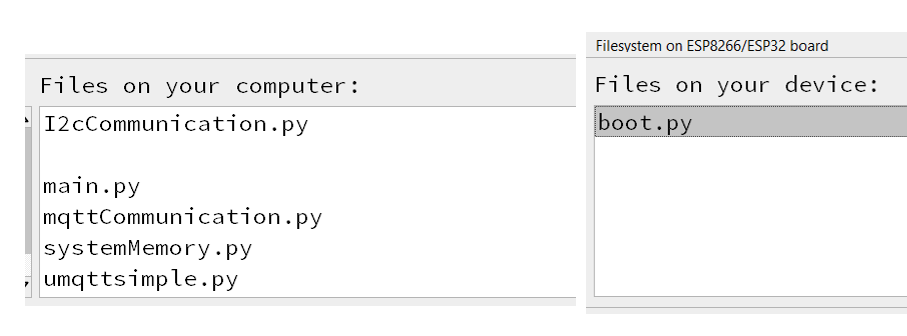


Figure 12: Mu Editor Menu before programming the PICO

Make sure the “Files on your computer list” shows all the files in the figure above. Those are the files needed to program the PICO. The naming of the files is self-explanatory:

* I2cCommunication.py : This code configures the parameters of the I2C communication between the PICO and the STM32.
* Umqttsimple.py: The library that enables mqtt protocol on PICO.
* mqttCommunication.py: This code defines the topics for publishing/subscription and is responsible for sending and receiving Mqtt messages between PICO and raspberry PI.
* systemMemory.py: defines where the sensor readings and actuator setpoints are stored in the memory.
* Main.py: the main code that orchestrates the functions described above.

Note: Make sure at this stage to remove all the files (Except boot.py) under “Files on your device” list. It should appear as shown in the figure above.

Now, simply drag and drop the files from “Files on your computer” to “Files on your device”. Eventually, it should look as shown in the figure below:

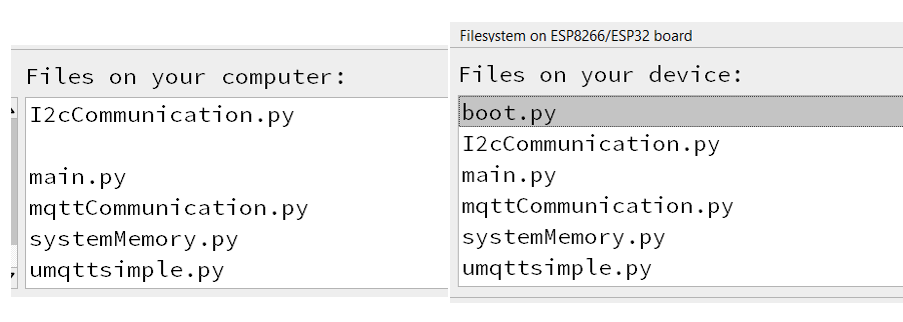


Figure 13: The python codes successfully transferred from PC to PICO

## Setting up the raspberry PI

### Simulink Support Package for Raspberry PI

As mentioned before, the raspberry PI runs the MPC algorithm. The algorithm is developed and deployed on the hardware using Simulink. Therefore, the raspberry PI operating system needs to be configured with the libraries and packages compatible with MATLAB/Simulink. This section explains how to install a customized Raspbian Linux operating system on Raspberry PI.

First, connect the raspberry pi to your network router using an ethernet cable as shown below. Make sure that your computer running Simulink is connected to the same network. Note that we need to use a wired connection between the router and the raspberry PI, because the wireless module on the raspberry PI is reserved for connecting to the local WI-FI network formed by the PICO.

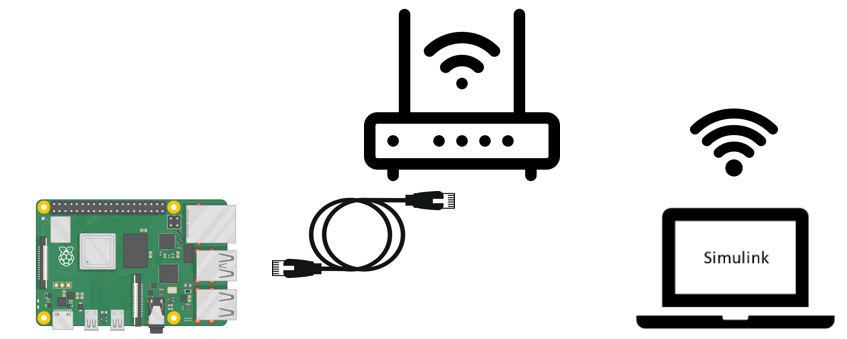


Figure 14: Connecting the raspberry PI to internet and Simulink

Next, in MATLAB main panel, go to “Add-ons” and select “Get Hardware Support Package”.

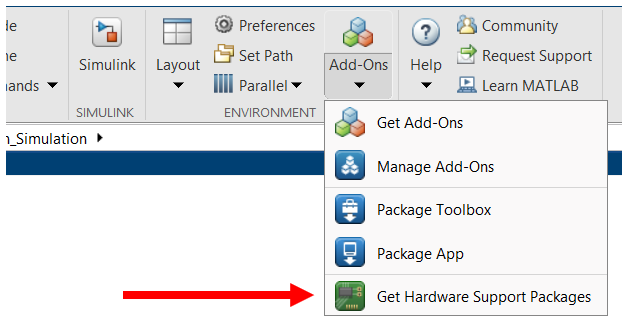


Figure 15: MATLAB Hardware support package menu

This will open the Hardware support package menu. On the search bar type “Raspberry PI” and select “Simulink Support Package for Raspberry PI Hardware”.

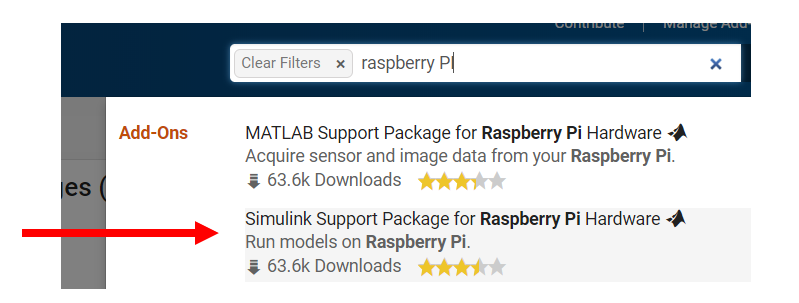


Figure 16: Simulink Support Package for Raspberry PI Hardware

Simulink Support Package for Raspberry Pi Hardware enables you to create and run Simulink models on Raspberry Pi hardware. The support package includes a library of Simulink blocks for configuring and accessing I/O peripherals and communication interfaces. It also enables you to interactively monitor and tune algorithms developed in Simulink as they run on Raspberry Pi.

This [video](https://nl.mathworks.com/videos/getting-started-with-simulink-support-package-for-raspberry-pi-1558342725160.html) by Mathworks provides a step-by-step guide to the download procedure. After following this procedure, your Raspberry PI is ready to run Simulink programs.

### Configuring the Raspberry PI Operating System

Connect the Raspberry PI to a screen using an HDMI cable. Connect a keyboard and a mouse using the USB ports. Then power ON the device.

* Set location, country code and language settings. The country code is important for WIFI access point.
* In the command terminal, execute the following command to ensure the system is up to date:
  + Sudo apt-get update
  + Sudo apt-get upgrade
* Install Mosquitto: Mosquitto is a message broker that implements several versions of the MQTT protocol. Recall that the raspberry PI uses MQTT protocol over WIFI to communicate with the sensors/actuators. To install Mosquitto, run the following command on the terminal:
  + Sudo apt-get install mosquito
* Setup the WIFI access point: The PICO microcontroller forms a WIFI network over which the MQTT communication takes place. Configure the Raspberry PI to connect to his network as follows:
  + Follow these [instructions](https://www.raspberrypi.com/documentation/computers/configuration.html#using-the-command-line) from Raspberry PI official documentation.
  + The Network details are as follows:
    - SSID : SIOM\_network
    - Password: [q<9LGSy4[
  + Make sure the host IP address is set to: "192.168.4.1"

# Testing the actuators, Sensors & OSEM Board

## 6.1 The GUI for testing:

By successfully executing the previous steps, the OSEM hardware is configured to receive sensor readings and manipulate the actuators. Before deploying the MPC controller, we need to verify that the communication and manipulation is saucerful.

For this purpose, a Simulink file named “Sensor\_actuator\_test” is created. This Simulink file can be found in the repository. This Simulink file is basically a GUI that has the interface shown in the figure below.

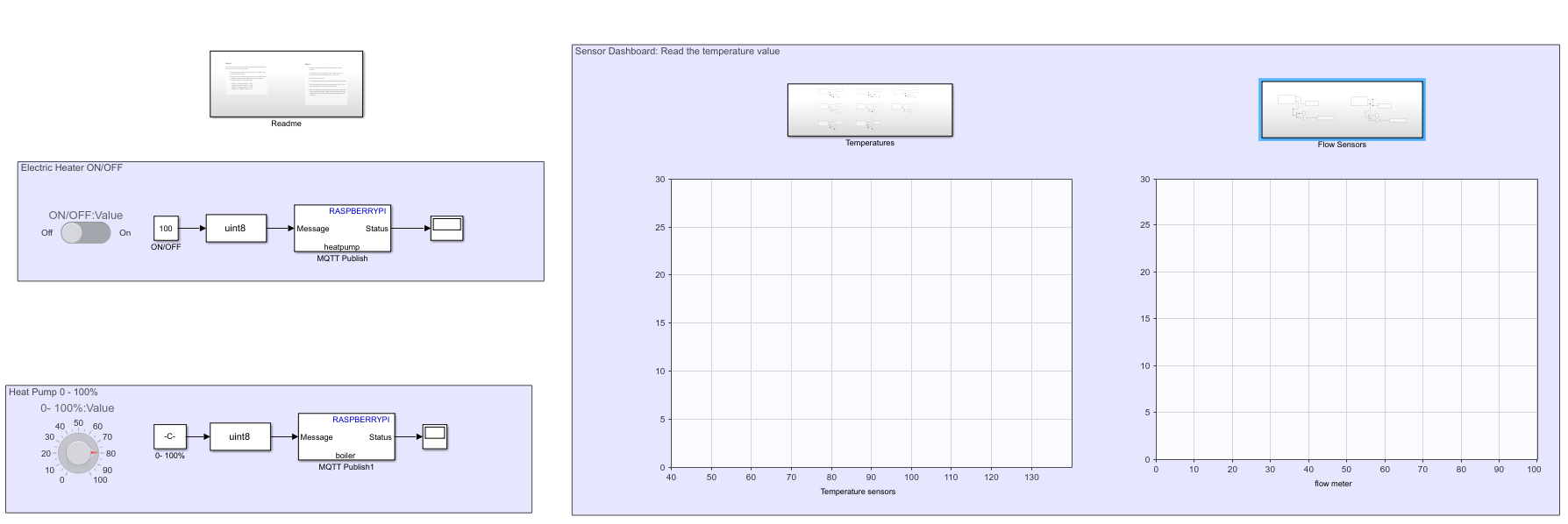


Figure 17: Simulink GUI for testing the system

On the left side of this GUI, a knob and a slider switch allow the user to manually manipulate the heat pump and the electric heater respectively. This is shown in the figure below.

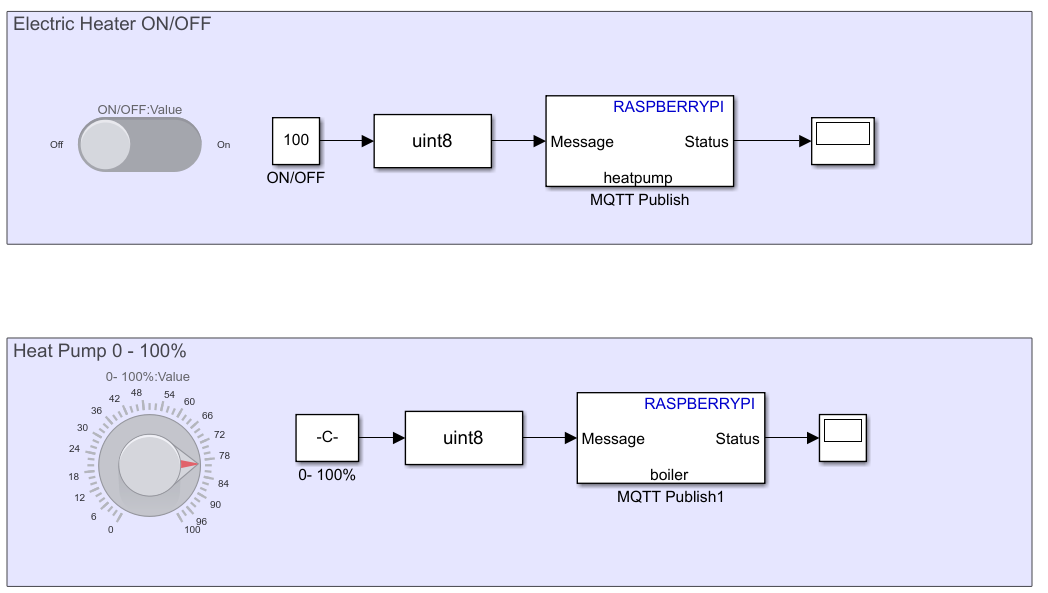


Figure 18: Manually Manipulating the actuators via Simulink (See note)

The following MQTT publish topics are valid:

|  |  |
| --- | --- |
| **Topic Name** | **Description** |
| valve1 | Analog output. Values from 0 to 100 are valid |
| Valve2 | Analog output. Values from 0 to 100 are valid |
| boiler | Analog output. Values from 0 to 100 are valid |
| heatpump | Digital output. 0 or 1 |

Figure 19: MQTT Publish topic names

**Important Note:** Due to a bug in the firmware/hardware connection, the heat pump setpoint and the electric heater setpoints are reversed. Meaning: the MQTT topic named “heatpump” will output only a digital output (0/1 Off/ON), while the MQTT topic “boiler” will output an analog value corresponding to 0-100%. This is the opposite of the actual system where the heat pump is modulated between 0-100% and the electric heater has an ON/OFF state.

An easy workaround to this bug is to also reverse the heat pump and boiler connections in the hardware. Meaning: Connect the heat pump to the boiler output port, and connect the electric heater to the heat pump output port.

To test any output port using this GUI, simply change the name of the MQTT topic in the MQTT publish block. The figure below shoes an example of sending an analog output signal via valve1 board on OSEM board.

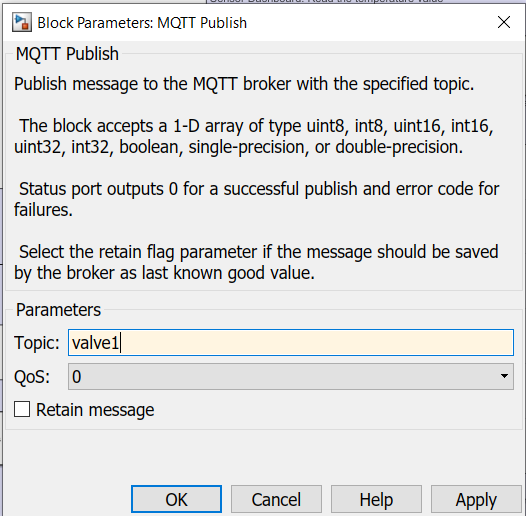


Figure 20: Manipulating outputs using MQTT publish block

In Summary, it doesn’t eventually matter where the heat pump and the electric heater are physically connected, as long as a compatible publish topic is used. For example, if it is desired to modulate the electric heater from 0-100%, it can be connected to the valve1 port, and the topic name on the GUI to be set to valve1.

The sensor values can also be observed on scopes in this GUI. 8 temperature sensors on the left graph, and 2 flow sensors on right graph. Reading the sensor values is implemented in a straightforward manner by subscribing to the relevant MQTT topics. An example of reading the temperature sensor connected to port TMP01 on OSEM board (See earlier figure) is shown below:

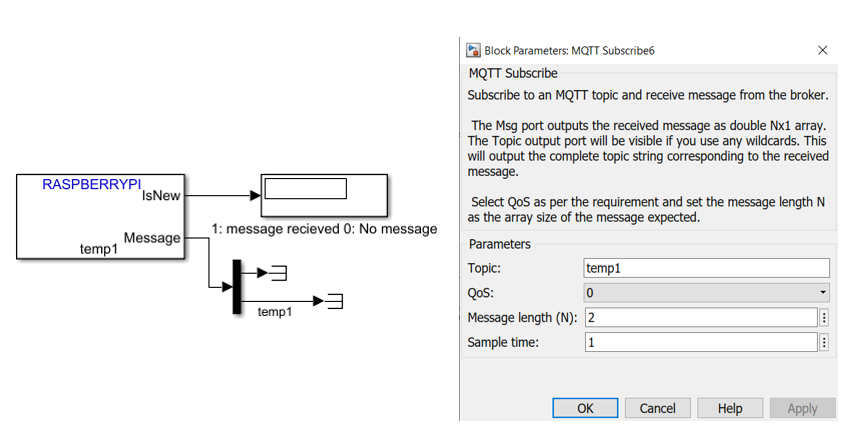


Figure 21: Reading the temperature sensor 1

Each sensor can be read using the MQTT subscribe block, with the corresponding topic name. The list of valid topic names is presented in the table below:

|  |  |
| --- | --- |
| **Topic Name** | Description |
| temp1 | Read temperature in [C] from TMP01 port |
| temp2 | Read temperature in [C] from TMP02 port |
| temp3 | Read temperature in [C] from TMP03 port |
| temp4 | Read temperature in [C] from TMP04 port |
| temp5 | Read temperature in [C] from TMP05 port |
| temp6 | Read temperature in [C] from TMP06 port |
| temp7 | Read temperature in [C] from TMP07 port |
| temp8 | Read temperature in [C] from TMP08 port |
| flow1 | Read flow rate in [Hz]\* from flow meter connected to port Flow1 |
| flow2 | Read flow rate in [Hz]\* from flow meter connected to port Flow2 |

Figure 22: Valid MQTT subscription topics

\*The flow meter reading can be converted from [Hz] to [l/min] with the following relation:

Where Q is the flow rate in [l/min] and F is the sensor reading in [Hz].

## 6.2 Running the GUI

Running the GUI will enable you to manipulate the actuators from Simulink and read the sensor data live. To run the GUI, follow the steps in this [section](#_Deployment_Procedure:). In the final step, click on “Monitor & Tune” and not on “Build, deploy & Start”.

# Deploying the MPC controller into the Raspberry PI Hardware

## The Simulink File:

Now that we verified the Communication between the raspberry pi and the sensors/actuators is working properly. We can deploy the MPC algorithm on the raspberry pi.

The MPC code that runs on the Raspberry PI is a C code that is autogenerated using Simulink. In this section, the Simulink file from which the code is autogenerated will be explained. This file is named “MPC\_deploy” and can be found in repository. In the next section, the procedure for deploying this file into the raspberry PI is detailed. The Simulink file is shown in the figure below:

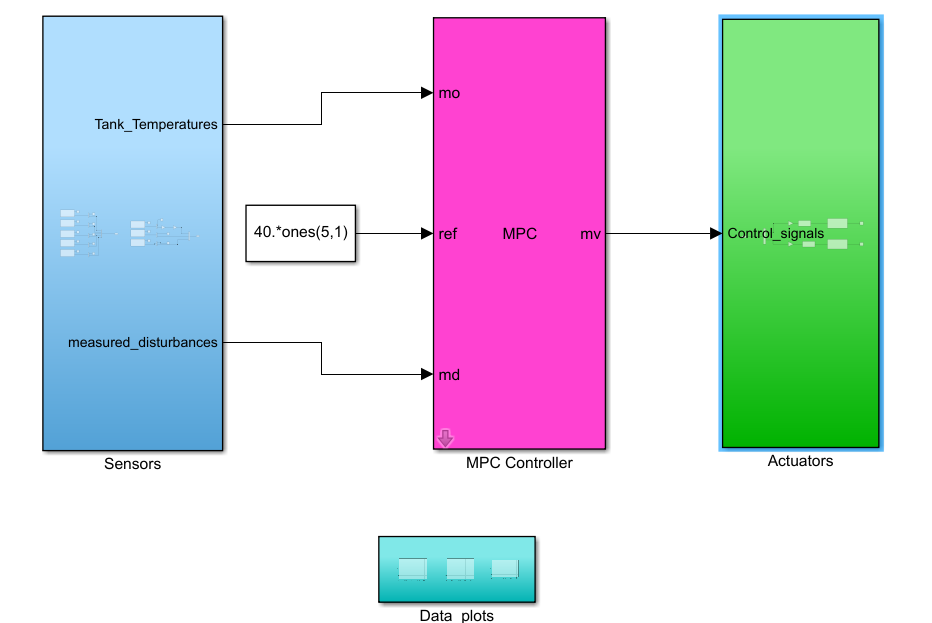


Figure 23: Simulink MPC diagram deployed to Raspberry PI

The Simulink diagram is clearly divided into 4 distinct subsystems:

1. Sensors:

The Sensors subsystem is shown in the figure below. The purpose of this subsystem is to collect the data from the temperature and flow sensors, and pass these sensors readings to the MPC block.

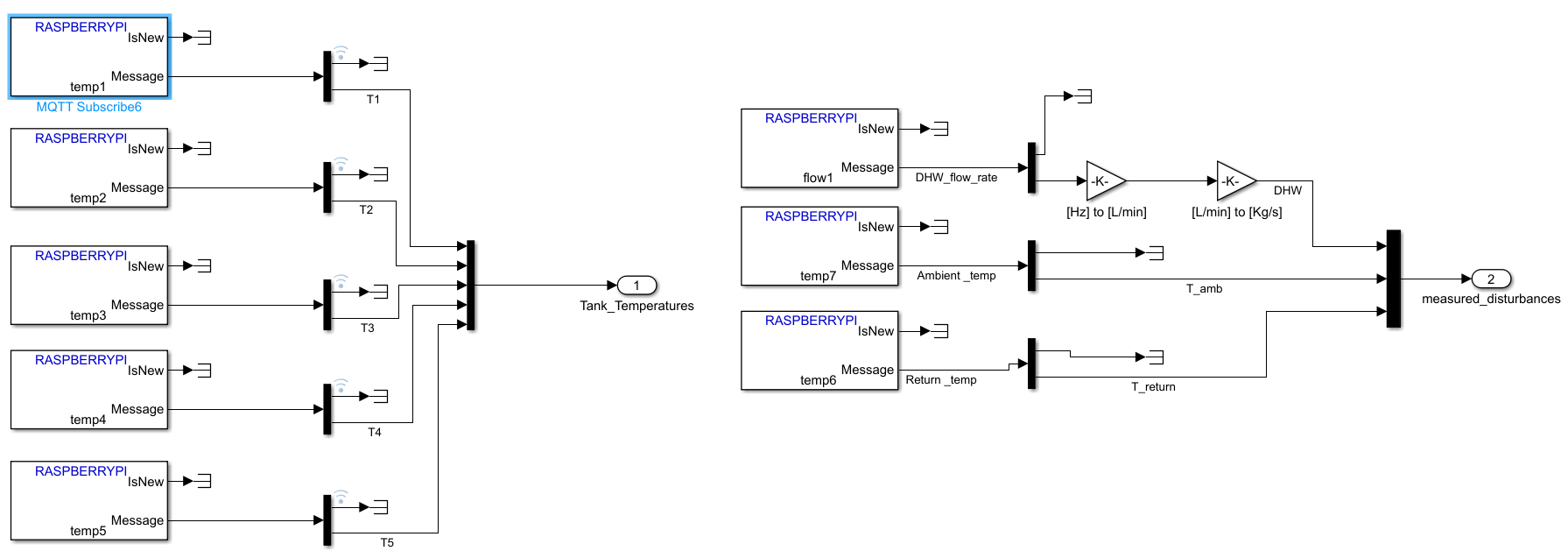


Figure 24: Sensor Subsystem

Each sensor is read individually using an “MQTT subscribe block”. Each sensor has a unique MQTT topic which is received every second using this block. The details for reading temperature sensor 1 is shown in the figure below. The other sensors follow identical structure except for the topic name.

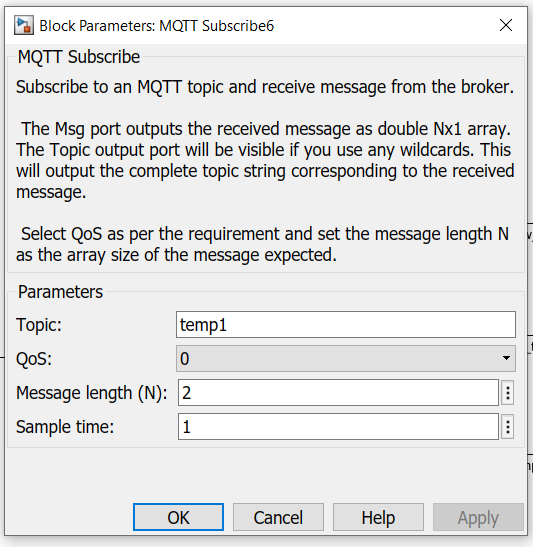


Figure 25: MQTT Subscription block for reading sensor data

1. MPC Controller block:

As the name suggests, this block takes for input the temperature and flow sensors readings and generate the adequate control signal for the heat pump and the electric heater according to the MPC algorithm. The details for designing and constructing this block has been extensively covered in a separate documentation (See WP4 Document).

1. Actuators:

This block is responsible for sending the control signals generated by MPC to the actuator hardware. The details of this block are shown below

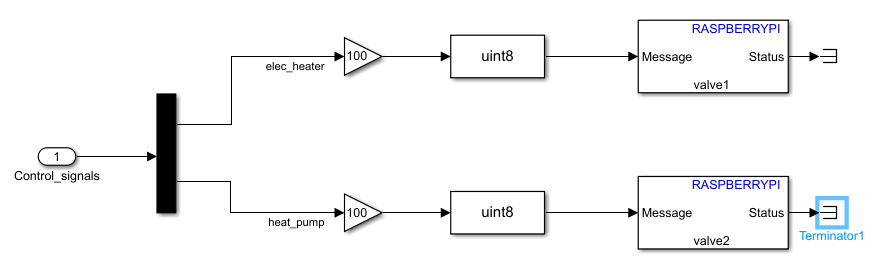


Figure 26: Actuator Subsystem

The control signals are sent using MQTT publish block. Each actuator has a unique publishing topic.

## Deployment Procedure:

1. Connect the Raspberry PI to PC: You can connect the raspberry PI directly to PC suing an ethernet cable. Or you can connect the RPI to your network router using an ethernet cable as show previously. In either case, make sure to note the IP address assigned to the RPI. You’ll need to enter this IP address later.
2. Open the Simulink file “MPC\_deploy.slx”. On the top panel, click on “Hardware” and go to “Hardware settings”. This will open the configuration window.

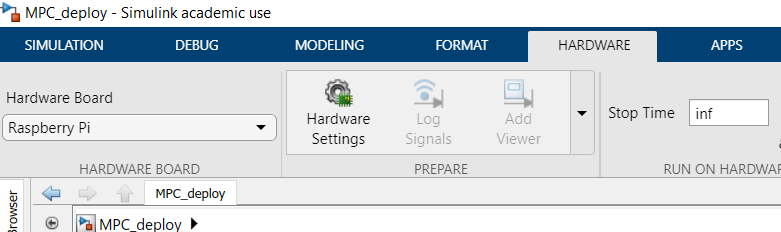


Figure 27: Accessing Hardware Settings

1. In the configuration window, click on “Hardware Implementation”. And from the “Hardware Board” drop down menu, select “Raspberry PI”

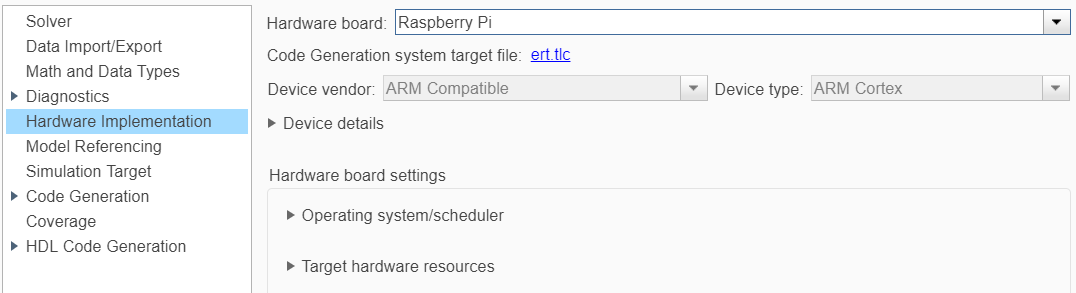


Figure 28: Selecting RPI as the target Hardware

1. On the same window, click on “Target hardware resources”, and select “Board Parameters”. This is where you provide Simulink with the credential that allow it to access the raspberry PI. Enter the IP address you recorded in step 1. Also, enter the username and the password of your Raspberry Pi operating system. Note that your own parameters will differ from the ones shown below.

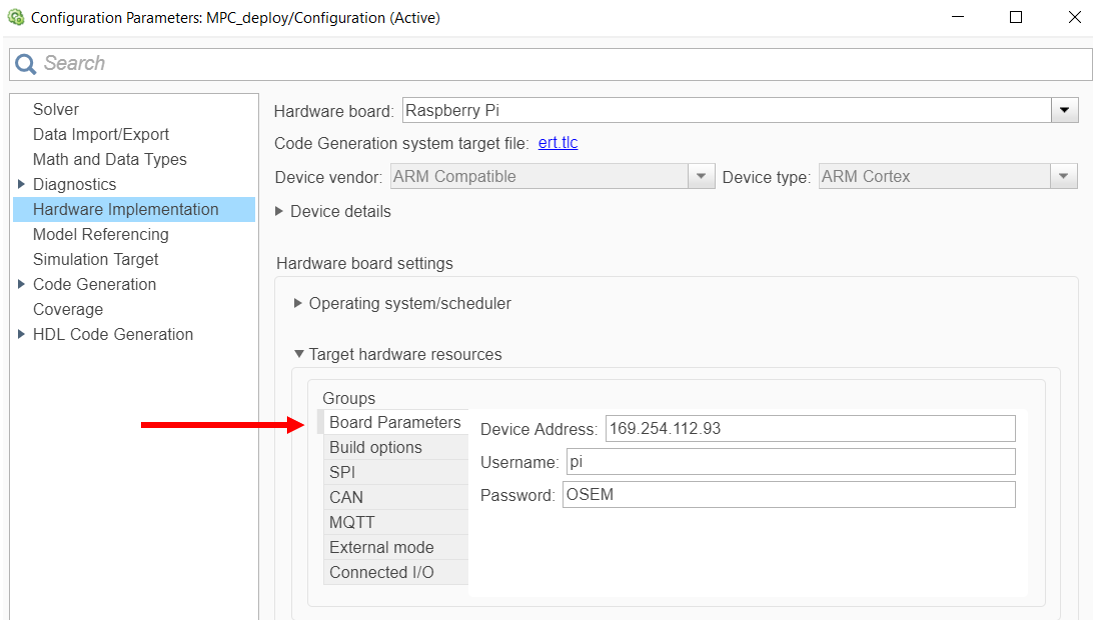


Figure 29: entering the RPI parameters

1. On the same panel, click on the “MQTT” and enter the following parameters:

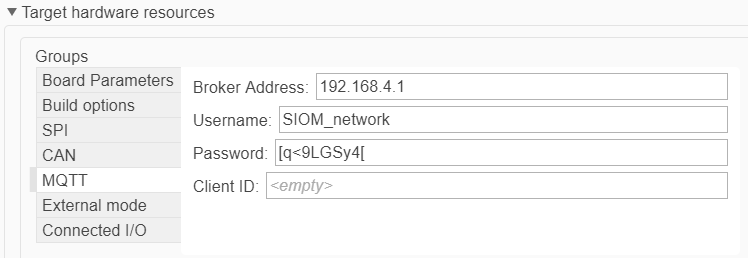


Figure 30: entering the MQTT parameters

1. Finally, on Simulink’s main pane, click on “hardware”. And click “Build, Deploy and Start ”. This will autogenerate a C code from the Simulink diagram, transfer it to the RPI and run the cod eon the RPI. Note that this process takes time, you can monitor the progress by observing the messages on the bottom left corner of your Simulink interface.

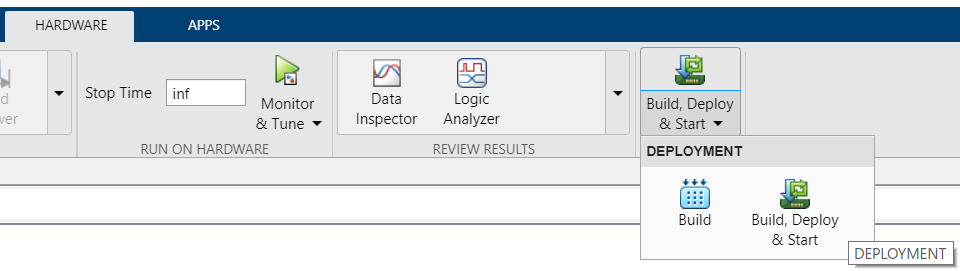


Figure 31: Deploying the code after configuring the settings