# Information about the EMLI project and the individual presentation portfolio

*Version v2023-04-20, Kjeld Jensen kjen@sdu.dk*

## Introduction

The course description presents the exam in EMLI as an “individual presentation portfolio submitted in the end of the semester” which will be graded according to the 7-point grading scale. This document contains further information about the EMLI project and the individual presentation portfolio.

## Course flow

The course is taught as a laboratory course. The first 9 modules follow a scheme of introductions to the theory followed by laboratory exercises performed in class in the course teams. No reports are submitted during those modules.

The modules 10-12 are reserved for an embedded linux project. At the 10th module a project description for the project is provided and introduced, and the students will start working on the project. During the modules 10-12 the teacher will be available in class for questions and support

## Project teams

The students will conduct the project and prepare the team contributions to the individual presentation portfolio in the existing course teams.

## Individual presentation portfolio submission deadline

The individual presentation portfolio must be submitted via SDU digital exam June 6th, 2023 at 10.00.

## Exam assessment

At the beginning of module 11, April 27th you will receive more details about the assessment process.

# Project assignment

In the EMLI 2023 project you will create an *embedded plant watering system* as described in this document.

The project assignment is to...

*in the team:*

a) Prepare the microcontrollers used for the embedded plant watering system

b) Design and develop an embedded linux system that fullfills all or as many as possible of the functional requirements and nonfunctional requirements.

*individually:*

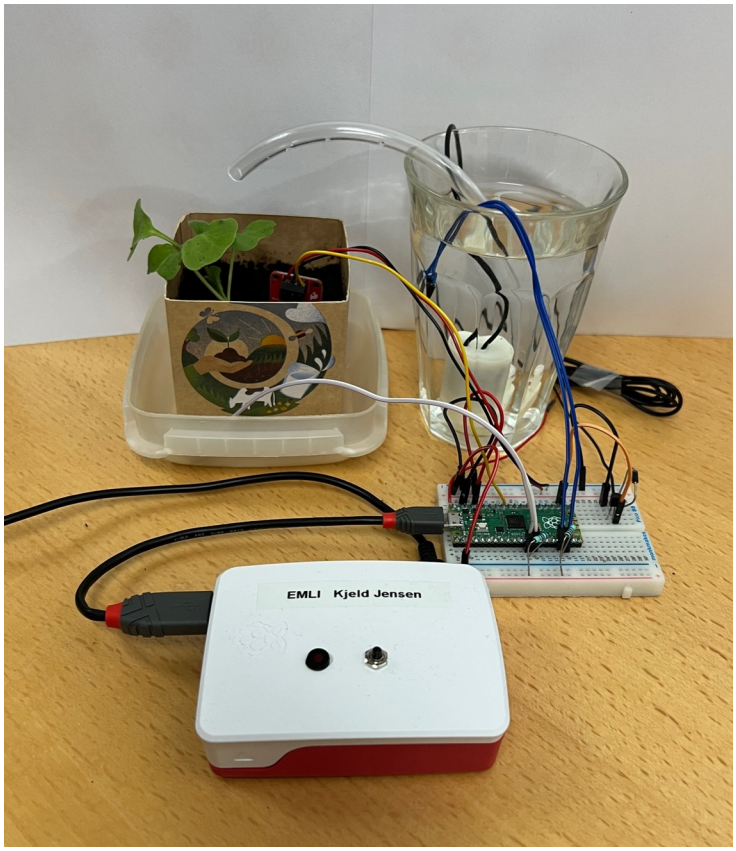
c) Submit an individual presentation portfolio describing the project in accordance with the listed requirements

The project focus is on the *embedded linux* part of the embedded plant watering system and you will therefore, like in the previous lab exercises, receive detailed descriptions and programming examples for the microcontrollers used.

Please notice that though the materials and information are provided, you do not have to grow plants (or try to do so) to complete the project successfully. But doing so will provide much more useful data.

## System description

The picture below shows the embedded system for watering crop plants. It consists of the EMLI kit Raspberry Pi 4 (RPi) connected to the EMLI kit Raspberry Pi Pico (Pico) microcontroller via USB. The Pico interfaces to all sensors and to the watering pump. The RPi controls the watering based on sensor input as well as input from the farmer who grows the plants.




The plant pot and water tank are made of common household materials. The pot is made by cutting the top off a milk carton and making some water drainage holes at the bottom of the carton. The soil has been taken outside.

## Project components

The EMLI kit does not contain all components required to complete this project. Each team will therefore receive a bag containing extra components.

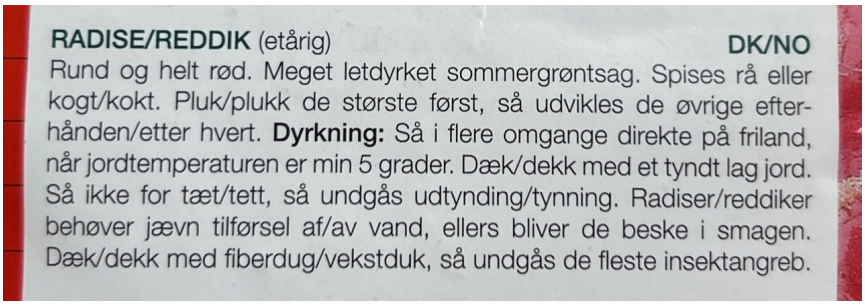
Please return the bag of components when returning the EMLI kit. Please make sure that these componets are returned in the seperate bag, i.e. not mixed up with the EMLI kit components.


### *List of project components*

| *Pcs.* | *Component* | *Reference* |
|---|---|---|
| 1 | Water pump | https://www.adafruit.com/product/4547 |
| 1 | Soil moisture sensor | https://www.sparkfun.com/products/13637 |
| 1 | LED red | |
| 1 | LED yellow | |
| 1 | LED green | |
| 3 | Resistor 220 Ohm | |
| 2 | Resistor 5.6 MOhm | |
| 1 | FET 2N7000 | https://www.st.com/resource/en/datasheet/cd00005134.pdf |
| 1 | Diode 1N4004 | |

## Plant growing

You will be provided a few seeds of radishes. They are easy to grow and sprouts within a couple of weeks. When seeding cover  the seeds with a thin layer of soil. Radishes need a regular supply of water in order not to get a taste of bitterness.

**RADISE/REDDIK** (etårig) **DK/NO**

Rund og helt rød. Meget letdyrket sommergrøntsag. Spises rå eller kogt/kokt. Pluk/plukk de største først, så udvikles de øvrige efterhånden/etter hvert. **Dyrkning:** Så i flere omgange direkte på friland, når jordtemperaturen er min 5 grader. Dæk/dekk med et tyndt lag jord. Så ikke for tæt/tett, så undgås udtynding/tynning. Radiser/reddiker behøver jævn tilførsel af/av vand, ellers bliver de beske i smagen. Dæk/dekk med fiberdug/vekstduk, så undgås de fleste insektangreb.

That's it for the crop growing tips, this not a critical, and you may choose to build the embedded system for watering plants and then document and demonstrate the functionality without actually growing plants.

# Pico sensor and pump control description

## *Sensors*

The Pico interfaces to all sensors and to the watering pump. The sensors are described below.

*Pump water alarm*

This sensor consists of two wires, one connected to 3V3 and one connected to GND via a large resistor (5.6 MOhm). When the two wire ends are open, the digital input is low (0), however when the wire ends are submerged in water, a small current flows between the wire, and the digital input goes high (1).

The sensor is placed in the water tank above the pump. When there is enough water, the sensor reports 1. When the water level drops below the sensor and the pump is thus at risk of running dry, the sensor reports 0.

You can test the sensor by shorting the two wires.

*Plant water alarm*

This sensor works like the *pump water alarm* sensor.

The sensor is placed in the water container below the pot. When the container is empty, the sensor reports 0. When water enters the container through the pot and there is a risk of flooding, the sensor reports 1.

You can test the sensor by shorting the two wires.

*Plant moisture sensor*

The plant moisture sensor has a signal out (SIG) which returns an anlog voltage between 0 and 3V3. This output is connected to ADC0.  In the example program below the moisture is represented by a percentage between 0 and 100.

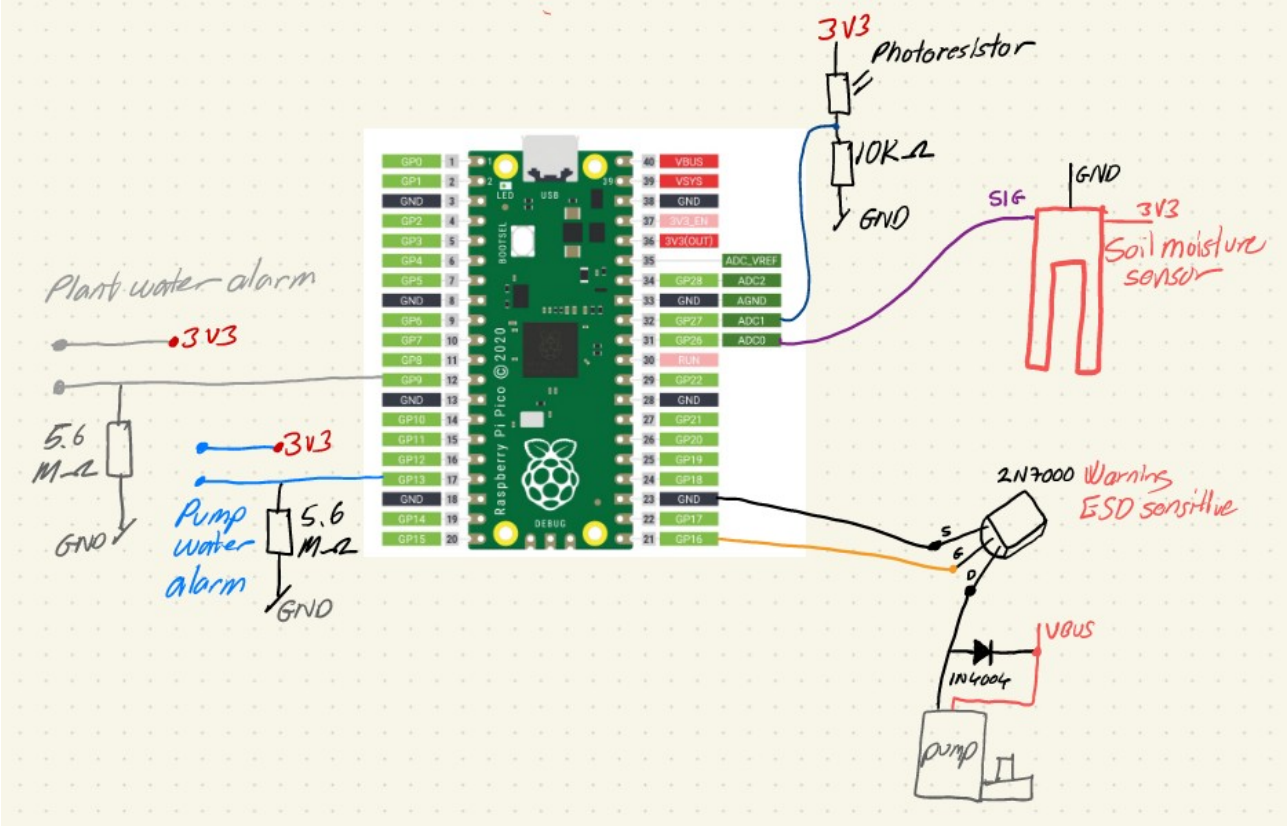
You can test the sensor by touching both "legs" of the sensor with your fingers.

*Ambient light sensor*

The ambient light sensor is constructed using the EMLI kit photoresistor as in one of the previous modules. In the example program below the ambient light is represented by a percentage between 0 and 100.

You can test the sensor by providing shade or pointing a light towards the sensor.

### *Schematics*




Important: Please notice that the FET 2N7000 is sensitive to static electricity. Try to handle it witout touching the pins directly and please don't sit on a wheeled chair while handling it.

Important: The water pump must be submerged in water while running, otherwise it may become defective.

### *Example program*

```
# Embedded Linux (EMLI)
# University of Southern Denmark
# Raspberry Pico plant watering controller
# Copyright (c) Kjeld Jensen <kjen@sdu.dk> <kj@kjen.dk>
# 2023-04-19, KJ, First version

from machine import Pin, ADC, UART
import utime
from sys import stdin
import uselect

pump_control = Pin(17, Pin.OUT)
pump_water_alarm = Pin(13, Pin.IN)
plant_water_alarm = Pin(9, Pin.IN)

moisture_sensor_pin = Pin(26, mode=Pin.IN)
moisture_sensor = ADC (moisture_sensor_pin)

photo_resistor_pin = Pin(27, mode=Pin.IN)
light_sensor = ADC(photo_resistor_pin)

led_builtin = machine.Pin(25, machine.Pin.OUT)

uart = machine.UART(0, 115200)

def moisture():
    return moisture_sensor.read_u16()/655.36

def light():
    return light_sensor.read_u16()/655.36

def pump_request():
    result = False
    select_result = uselect.select([stdin], [], [], 0)
    while select_result[0]:
        ch = stdin.read(1)
        if ch == 'p':
            result = True
        select_result = uselect.select([stdin], [], [], 0)
    return result

while True:
    led_builtin.toggle()
    if pump_request():
        pump_control.high()
        utime.sleep(1)
        pump_control.low()
    else:
        utime.sleep(1)
    print("%d,%d,%.0f,%.0f" % (plant_water_alarm.value(), pump_water_alarm.value(), moisture(),
light()))
```

The example program above has been written for the Pico to read sensor values and control the pump. You may use the program as is or you may modify it for your design.

The Pico LED blinks each second to signal that the program is running.

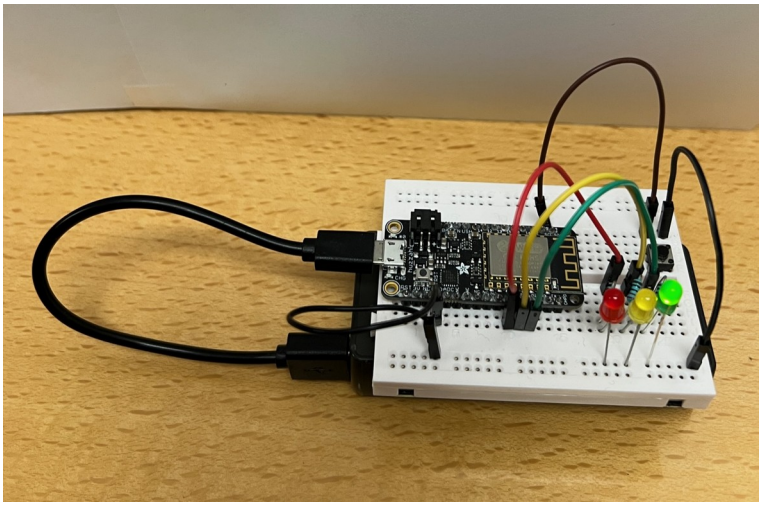
The program listens for the character `p` on the serial port. If received, the pump runs for 1 second. Other characters are disregarded.

The program sends sensor values as a comma seperated string to the serial port each second. The the format is:

`plant_water_alarm,pump_water_alarm,soil_moisture,ambient_light`

## ESP8266 wireless remote description

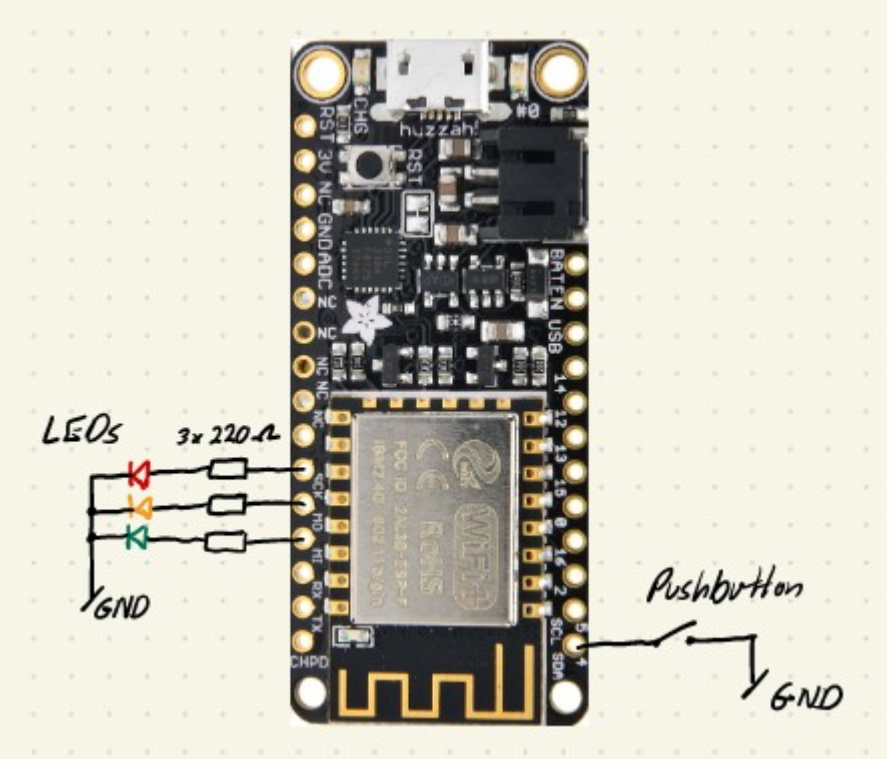
The farmer has a wireless remote to monitor the status of the plant watering system and to activate the water pump. It is based on the EMLI kit Adafruit HUZZAH ESP8266 microcontroller (ESP8266) and is illustrated in the picture below.

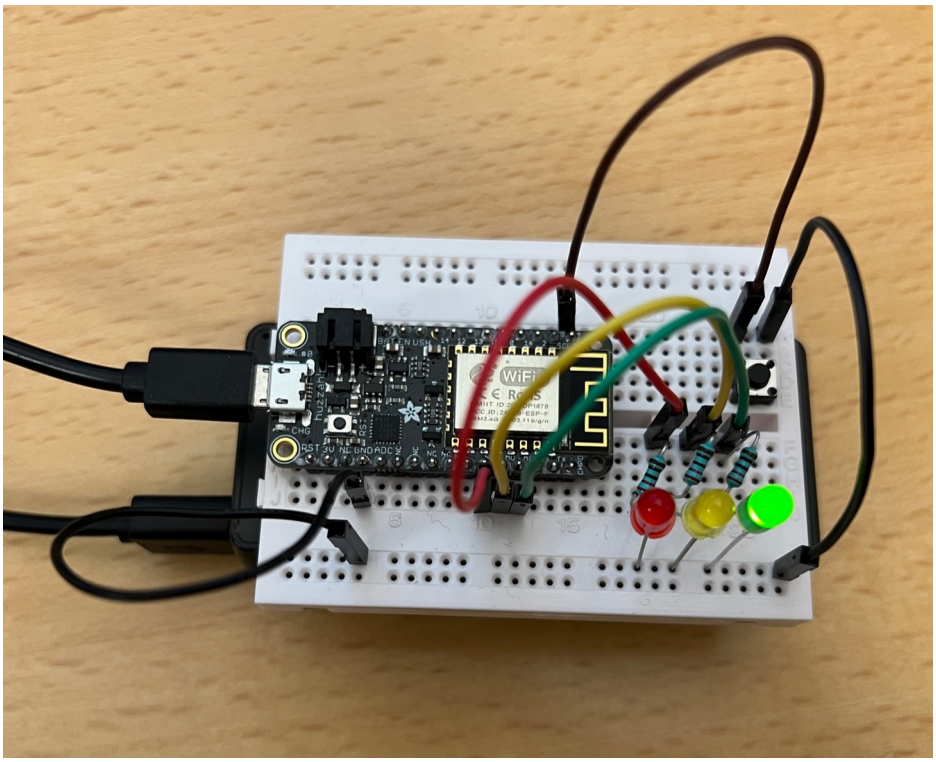


The picture shows the ESP8266 powered by a USB power bank (below the breadboard). You are welcome to power the ESP8266 by your computer or a USB power supply instead.

The ESP8266 is connected to three LEDs (red, yellow, green) and a single push button as detailed in the schematics below.

***Schematics***



### *Example program*

An ESP8266 example program is available on itslearning. You may use the program as is or you may modify it for your design.

The program connects as a client to a Wifi access point using a defined SSID and password.

The ESP8266 builtin LED turns on when powered up/reset. It turns off when the ESP8266 has successfully connected to the access point.

The program is configured to use a static IP address (default is 192.168.0.222) instead of obtaining an IP address via DHCP. This is to make it easier to access the webserver described below.

When connected to the access point The program runs a webserver that provides access to the wireless remote functionality via the GET requests listed below:

```
http://192.168.0.222/led/red/on
http://192.168.0.222/led/red/off
http://192.168.0.222/led/yellow/on
http://192.168.0.222/led/yellow/off
http://192.168.0.222/led/green/on
http://192.168.0.222/led/green/off
http://192.168.0.222/button/a/count
```

## Functional requirements

The functional requirements for the embedded plant watering system are:

a) The water pump must be controllable

b) The described sensors must be readable

c) The wireless remote functionality must be accessible

d) The water pump must run once at an interval of 12 hours.

e) The water pump must additionally run maximum once per hour if the soil moisture falls below a certain threshold

f) The water pump must run once if the farmer presses the button once within a period of two seconds.

g) In case of an active pump water alarm or a plant water alarm the water pump may not run under any circumstances.

h) The wireless remote must light the red LED in case of a pump water alarm or plant water alarm.

i) The wireless remote must light the yellow LED in case the soil moisture is below a certain threshold

j) The wireless remote must light the green LED otherwise

k) The RPi must provide web access to historical information about soil moisture, ambient light, pump activations and water alarms. The data should be available as both graphs and downloadable data

l) The RPi must be able to function both with and without an active internet connection via the RPi ethernet port

m) The RPi must become fully operational after at power failure without requiring user interaction

n) The RPi system and internet performance and health must be monitored and logged

## Nonfunctional requirements

a) The embedded system must be designed with the main tenets of the Unix philosopy in mind

b) Where appliable shell scipts should be used rather than other programming or scripting languages

c) Either MQTT or ROS2 must be used as message passing system for the embedded linux system. All sensors, the pump actuator, the wireless remote functionality, the logic control etc. must be represented as message topics

d) Where applicable the embedded system must be scalable towards handling multiple plants (disregarding the limitation of serial port interfaces).

e) The RPi must to the extent possible be secured against malicious cyber attacks from both the internet connection, when active, and the local wifi

## Individual presentation portfolio

The individual presentation portfolio contains:

1) **A report describing the project.** Report pages must be A4 format and have 2 cm margins. Any pages beyond the stated maximum number of pages will not be read. Add images, sketches etc. as applicable. Page numbering must be present but no other headers or footers on the pages. Font must be Times New Roman, Liberation Serif or similar at 12pt. The report consists of two individual pdf documents described in the following:
   a) **a team document (11 pdf pages)** following the structure:
      - Front page: clearly stating project group number, group member names and SDU email addresses. No other report content at this page.
      - Solution approach: Describe the high level solution approach, the overall architecture and design, including advantages and drawbacks. Use sketches and diagrams.
      - Solution description: Describe solutions to key tasks and concepts in the project.
      - Tests and results: Describe how the embedded system was tested and present the results of those tests.
      - Conclusion: Short conclusion summarizing the achieved solution and results.
   b) **an individual document (4 pdf pages)** following this structure:
      - Front page: clearly stating student name and SDU email address. No other report content at this page.
      - Discussion: Compare the achieved solution, results, demonstration to the functional requirements. Discuss to what extent the solutions achieve the nonfunctional requirements, and if not, why not.
      - Conclusion: Summarize and outline how the achieved solution and results can be improved in future work.
2) **A team video** presenting the outcome of the project:
   a) The video will be a combination of illustrations, recorded video of the physical system and any interactions, relevant screen recordings etc.
   b) The video length must be maximum 90 seconds.
   c) The video must be formatted as MP4 encoded using H.264/AAC with a maximum file size of 25 Mb. If not, the video may not be included in the assessment. Use of `ffmpeg` to convert to this format is recommended, the following parameters appears to work well for different sizes and formats:
      - `ffmpeg -i input.mov -c:v libx264 -vf scale=-1:1080 -crf 23 -maxrate 2M -bufsize 4M output.mp4`
3) **A link to a git repository** containing relevant scripts, configuration files etc. created by the team for solving the project.
   a) The git repository must provide a quick and intuitive access to the relevant material.