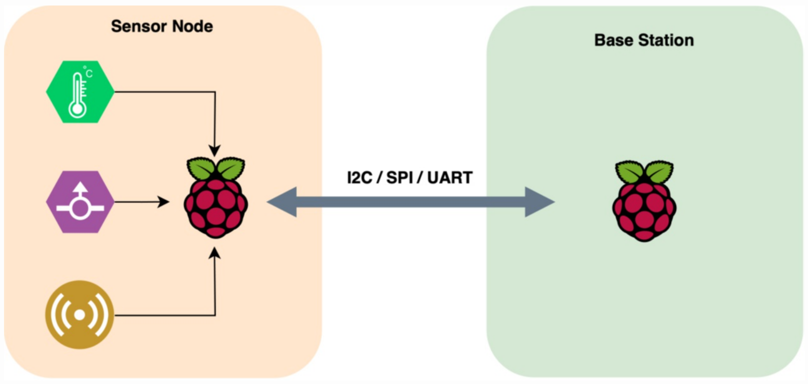
Project Sensor Node

As embedded software developers we often work directly with different hardware peripherals such as sensors and actuators. In this project, you will learn how to develop a sensor node, i.e. a collection of sensors that you can communicate with through a common interface. You will also learn how to develop a low-level driver for your sensor node for the Zephyr RTOS.

Introduction

The goal of this project is to develop a smart sensor node that incorporates multiple different sensors. You will have to develop both the hardware platform (on a breadboard), and the interface to your sensor node. The overall system should look something like this:



Multiple sensors are connected to a RP Pico, which acts as the communication interface with other controllers. You have a lot of freedom in selecting the particular sensors you want to use. You should try and aim for a particular use case, such as an environmental data collection unit (temperature, pressure, humidity, CO2 concentration, …), a smart home unit (temperature, light, noise level, …), or some form of home security device (noise, vibration, distance sensors).

Both the base node and the sensor node will run on RP Pico boards.

For the base node, you will need to write a Zephyr sensor driver that provides an interface to the sensor node by implementing the Sensors API. Therein, you are free to choose between implementing the Fetch and Get API or the more recent Read and Decode API. You will also need to write an application which you will use to demonstrate the driver's functionality (check Zephyr codebase for examples).

For the sensor node, you have freedom in your choice of software stack. For example, you can develop your firmware on directly on top of Pico SDK, or you can use FreeRTOS or Zephyr. Note that if you use Zephyr, you are not allowed to use Sensors API to interact with individual sensors. You can, however, use I2C, UART or other communication protocols (as you did in Lab 3 part 1).

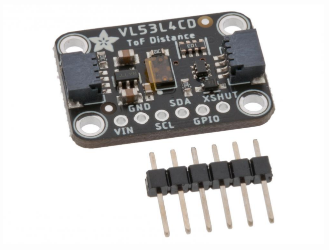
Hardware Requirements

You first have to decide on your sensors. You have a lot of freedom here. The only requirements from our side are:

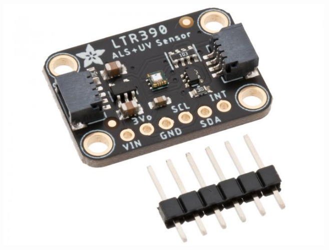
- you must include at least one analogue sensor
- the sensor must be availabe to order from Electro:Kit, ELFA, or Farnell, with

Electro:Kit being our preferred supplier
- the sensor must be feasible to use for prototyping, i.e. either it already comes with a breakout board, or it is easy to use on a breadbord without the need for soldering
- the combined cost should be reasonable (preferably under 1000 SEK)

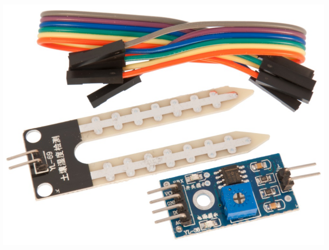
Here are some examples of sensors you may include:



Distance sensor: Electro:Kit



UV light sensor: Electro:Kit



Soil moisture sensor: Electro:Kit

When selecting sensors, you have to make sure that the sensors you select work with the 3V voltage levels the RP Pico can tolerate!

Project outline

There are multiple steps involved in this project. We will outline some of the most important ones here, however, it is up to each team to come up with a project timeline!

1. Defining node platform

You have to decide on the sensors you wish to use in your setup. You will need to consult the various datasheets of the sensors to figure out which work well with the RP Pico (e.g., share the same logical voltage levels), and what additional material you may need (each sensor may have a specific minimal working circuit it needs to function correctly). We don't expect you to create a custom PCB, but you should be able to draw the general circuit diagram of your sensor node.

At this step you also should already think about the software platform you want to use for the sensor node. You can either write your sensor node firmware directly on top of the Pico SDK, or you may choose to use Zephyr or FreeRTOS, the latter which can be run on top of Pico SDK. Note that if you use Zephyr, you are not allowed to use the Sensors API to interact with individual sensors.

1. Define SW protocol

In order to add a digital interface to your sensors you need to agree on a protocol for accessing sensor data and possible configurations of the sensors. You can use your experience from Lab 3 of 1DT106 regarding what a possible I2C interface for a sensor can look like. Defining the protocol at this stage will also allow for the co-development of the sensor HW/SW and the Zephyr driver in parallel.

3. Create prototype board

Using the HW diagram from step 1 you will design a simple prototyping platform for the sensor node on a breadboard.

4. Write sensor node firmware and driver code

At this stage you can start implementing the firmware for the sensor node and the Zephyr driver. Try testing each of them in isolation as well.

5. Integrate + Test

You should design and employ a proper testing strategy for the sensor node you developed. This will depend on the particular sensors you have chosen. You need to perform system-level testing, and are encouraged to explore finer forms of testing (e.g., unit testing). You also have to develop a Zephyr example application that showcases how communication with your sensor node works.

Deliverables

Deliverables will be outined on Studium.

Grading

The grade of your project depends on multiple factors:

For a passing grade we expect the following: - a working setup with at least 2 sensors - a basic Zephyr driver that can read out values from both sensor channels - a small Zephyr example illustrating your solution

To achieve a higher grade we expect at least the following:

For a 4, you should incoporate a third sensor, and allow for some configuration of the sensors through your protocol (e.g. setting different ranges, precisions, oversampling rates, …).

For a 5, you should incorporate an interrupt line, that can be set through configuration registers in your node interface, and incorporate a small example in Zephyr on the RP Pico showcasing that interrupt. It is up to you to decide what kind of interrupt service you want to provide (e.g. if you have included a temperature sensor, you might want to generate an interrupt signal whenever you reach a certain temperatue limit). You also need a thorough testing strategy that should be well documented in your report.

You will have the chance to present your system at the end of the course. Our grading decision will include the presentation, the report, and your code.

We will take additional factors into consideration during grading, including but not limited to the quality of the report and code, design choices such as the particular software stack used, etc.