[ELE00118M] MSc Practical Robotics (PRAR)

[ELE00103M] Year 4 Robotics module

Lab Session 2 Handout

Lab 2: Embedded Navigational Sensing

Lecturer: Mark A Post (mark.post@york.ac.uk)

Demonstrator: Robert Woolley (rw1445@york.ac.uk)

Technician: Mike Angus

1. Aims and Objectives

In this two-hour laboratory session, you will continue the work from the previous lab, adding a display to the Raspberry Pi and sensors to measure distance on your robot.

2. Learning outcomes

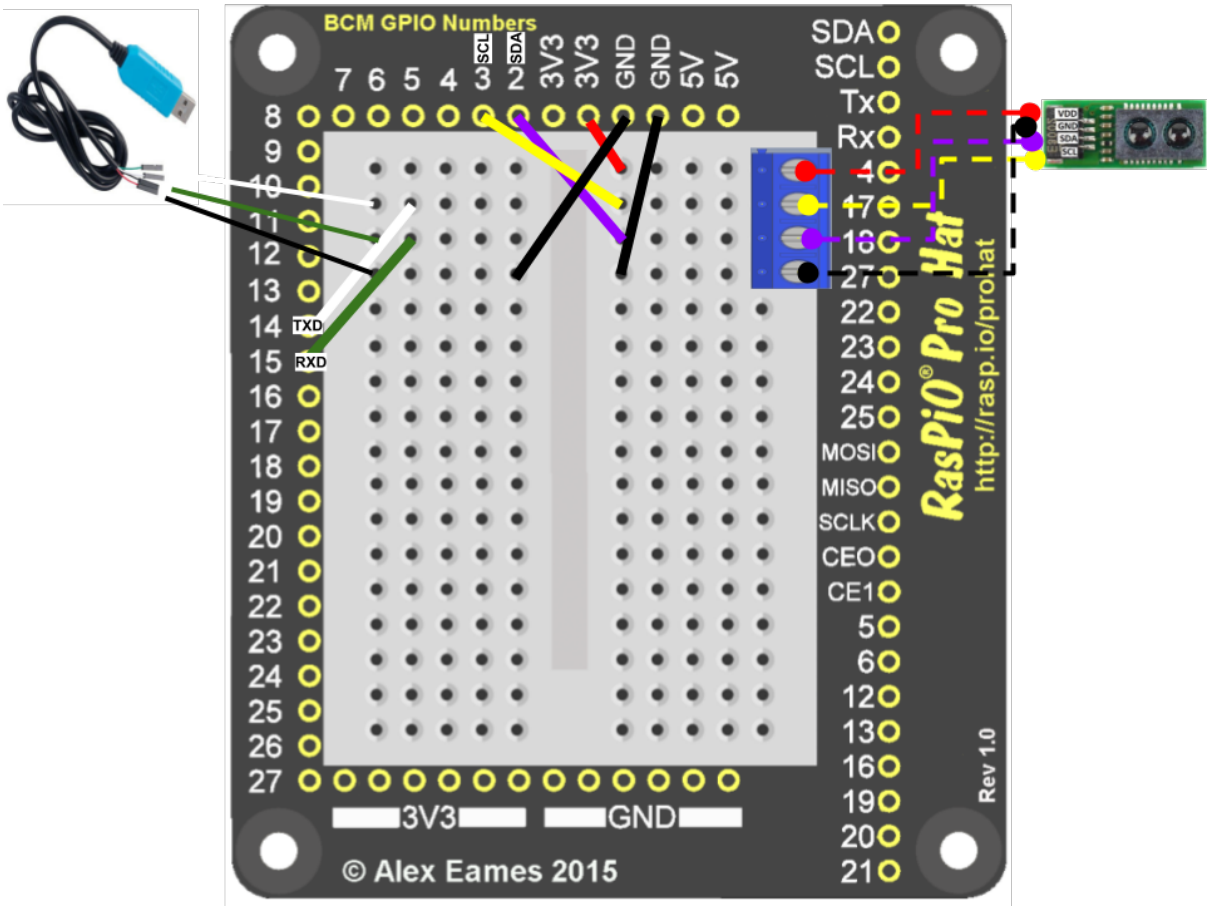
- Reading sensor data in to a microcontroller
- Displaying sensor data
- Communicating these readings to other devices

3. Software and hardware

THE RASPBERRY PI PLATFORM

You will use C or Python programs using the Raspberry Pi from Lab Session 1 connected to either the USB-Serial connector which we use to attach it to a PC or a network connection via SSH.

The GPIO pins on the Raspberry Pi will be used to communicate with the sensors. Remember that the maximum current a pin can sink or source is 50mA, so choose appropriate resistors and use a buffer-circuit if needed. While assembling parts on your robot please refer to the “Wiring Instructions” section starting on page 50 of the document “**MEng/MSc Robotics Kit**” for guidance and tips on how to build a good quality robot.



4. Pre-Lab Preparation

- Have at least a cursory read through this lab script from beginning to end to familiarize yourself with what you will be doing in this practical exercise.
- Look up the datasheet for the Sharp GP2Y0E02B infrared range sensor, and familiarize yourself with how to connect it to the Raspberry Pi using an I^2C pin and how you will convert the read value into a calibrated distance.

NOTE: Make sure to back up your work frequently from the Raspberry Pi, at least every lab, as small errors can make the SD card unreadable and destroy files!

5. Tasks

In the lab we have a number of different distance sensors with either use infrared light or ultrasonic sound waves as their medium. The goal of this lab is to interface with one of these sensors using your Raspberry Pi and create a digital distance measuring device.

TASK 1: CONNECT A SENSOR

First, connect the Sharp GP2Y0E02B to the I2C pins of the Raspberry Pi. A recommended wiring diagram is above. You will need to find (online) the data-sheet for the device and figure out how to use it to get a distance measurement. Now that you are familiar with connecting to the Raspberry Pi, it is a good time for you to get used to looking up online resources for making devices like the Sharp GP2Y0E02B work. I am not putting detailed instructions here as the goal is for you to get used to working with datasheets and other online resources to get devices to work. Using example code found online as a guide is fine. As the GP2Y0E02B is an I^2C device, look for Raspberry Pi tutorials and code that others have written in Python or C that show how to use the I^2C bus. You may for convenience want to use the I^2C functions available in the pigpio library: http://abyz.me.uk/rpi/pigpio/cif.html#i2cOpen for C and http://abyz.me.uk/rpi/pigpio/python.html#i2c_open for Python.

To make sure that the I2C sensor is working use the following console command:

```
i2cdetect -y 1
```

Then check if you see the number "40" in the table that is printed, that (as hexadecimal, 0x40) is the address of the sensor. If the table is printed very slowly or an error is printed then there may be a problem with your wiring or your sensor.

TASK 2: READ THE SENSOR

Make your program read the distance from the device at an appropriate rate, convert the distance into metric units and send the device as a string over the PC serial interface. You should use the “%f” format specifier in the print statement to the serial port to ensure that you are printing a precise floating-point number as you will need to calibrate the sensor to produce a measurement in practical units. Using internet searches you should be able to find the datasheets and fully-working code examples such as:

Datasheet: https://robot-electronics.co.uk/files/gp2y0e02b_e.pdf
Application Note: https://robot-electronics.co.uk/files/gp2y0e02_03_appl_e.pdf
Arduino C code example: https://robot-electronics.co.uk/htm/arduino_examples.htm
Python code example: https://github.com/GeeAlex/PiSharpIR

It is up to you to choose the references and example code that is suitable, and when you do, spend some time looking through the source in detail and figuring out how it works.

TASK 3: CHARACTERIZE THE SENSOR

It's essential to understand how a sensor behaves on a robot. Take several measurements with an obstacle in front of the sensor and try to give reasonable answers to these questions:

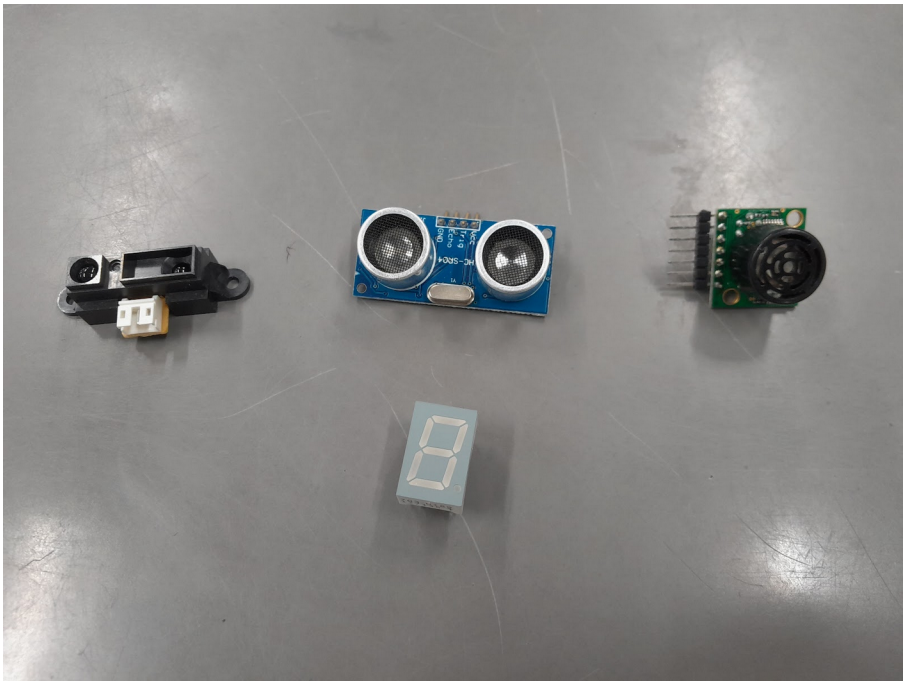
- How accurate is the device measurement?
- How fast does it refresh/take measurements?
- How wide is its field of view?
- How reliable does it seem?

Using a sheet of white paper as a reflective source, create a function which estimates the distance that the sensor is from the paper and prints the resulting distance. The sensor should naturally produce readings in cm. However, you should confirm this by plotting the observed value for the reflected IR strength at a number of distances, and then estimating the line that fits this data to a value.

Estimate how long it takes to perform the calculation (hint: use a timer and a for-loop to count how long it takes to perform the calculation several times to get a more accurate answer). Is this a problem? Often for performing tasks like this the most efficient method is to use a look-up table, where preset values are stored in memory and simple interpolations between the values are chosen. How accurate is your distance measure? What is the furthest distance you can estimate? What happens when we place the sheet beyond this distance? How well does it work on different sensors? How noisy is it?

EXTRA TASKS: OTHER SENSORS

There are a range of other sensors available in the lab. They are located in the boxes at one end of the room. Try finding them, look up the data sheet and get them to work. Although they are unlikely to be used on the final robot for the assessment it is a good experience to get them working.



Three possible sensors; sharp IR and two ultrasonic sensors. Finally a 7-segment display.

Once you have it working try to display the data on one or two 7-segment displays. Make sure you choose a resistor/s to not damage the raspberry pi or the 7-segment display.

EXTRA TASKS: PROCESS SENSOR DATA

Robots generally need to use more than one kind of sensor and process the data received further to make it more reliable. Averaging is one way to do this, or discarding extreme values that are obviously incorrect. You will need to improve the sensor data intelligently in your program by processing. Think about how you would like to do this and implement something that makes sense to you. This could be as simple as averaging each pair of sensor measurements together, or a more complex method such as finding the mean and standard deviation of a sensor and estimating the maximum likelihood. Remember, you may need to calibrate the two sensors separately to ensure that they give the same reading when faced with the same environment. Compare the performance of the processed sensor data with the raw data you characterized before:

- How accurate is the processed measurement?
- How fast does the whole system refresh/take measurements?
- How reliable does it seem?