A PharoThings Tutorial

Allex Oliveira

January 3, 2019

Layout and typography based on the `sbabook` LaTeX class by Damien Pollet.

Contents

Illustrations

CHAPTER 1

Lesson 8 - I2C Sensors (Temperature, Humidity, Pressure and Acellerometer)

In the previous lessons, we learned how to control LEDs and to use a button to interact with LEDs. Now let’s start using some sensors to interact automatically with the real world, taking the temperature, air pressure and humidity. This kind of sensor use the I2C protocol to communicate.

1.1 What we need?

In this lesson we will use a very simple setup.

Components

- 1 Raspberry Pi connected to your network (wired or wireless)
- 1 Breadboard
- 1 ADXL345 accelerometer sensor
- 1 BME280 temperature, humidity and pressure sensor
- 1 MCP9808 temperature sensor
- Jumper wires

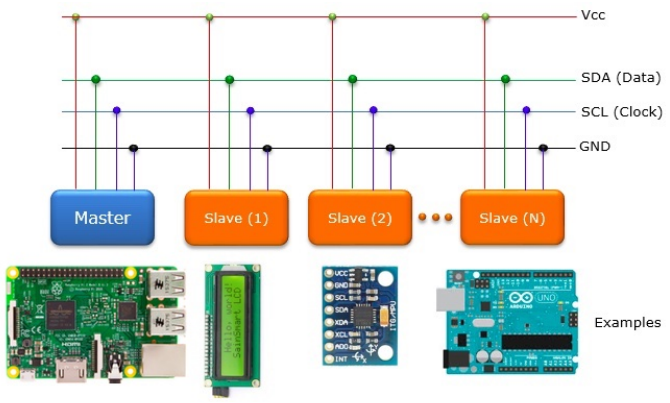

Figure 1-1 Devices connected using I2C bus.

1.2 Experimental theory

Before constructing any circuit, you must know the parameters of the components in the circuit, such as their operating voltage, operating circuit, etc.

The I2C protocol

The I2C communication protocol can be easily implemented in many electronic projects, being a very popular and widely used protocol. It is possible to perform communication between one or more master devices and several slave devices. It is an easy-to-implement protocol because it uses only 2 wires to communicate between up to 112 devices using 7-bit addressing and up to 1008 devices using 10-bit addressing.

The Figure 1-1 shows how you can connect the devices using the same I2C bus.

How I2C works?

How can we communicate with multiple devices using only two wires? For this to happen, each device has set an ID or an unique address. Then the master device can choose which device to communicate with.

The two wires are called Serial Clock (or SCL) and Serial Data (or SDA). The SCL wire is the clock signal that synchronizes the data transfer between devices on the I2C bus and is generated by the master device. The other wire is the SDA line that carries the date.

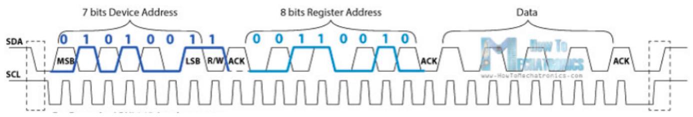

Figure 1-2 Bits sequence.

Protocol

The data is transferred in 8-bit sequences, like you can see in Figure 1-2. After a special starting condition occurs, comes the first 8-bit sequence that indicates the address of the slave to which the data is being sent. For example, for the ADXL345 accelerometer device, the default address is 16r53 (0X53) or 0101 0011 (the last bit actived means the device is on read mode).

After each 8 bit sequence follows a bit called Acknowledge. After the first Acknowledge bit, in most cases another addressing sequence comes, but this time to the internal registers of the slave device.

The internal registers are locations in the slave's memory containing various information or data. For example, the ADX345 accelerometer has a unique device address (16r53) and addition internal record addresses for the X, Y, and Z axes (16r32, 16r33, 16r34, etc.). Therefore, if we want to read the X-axis data, we first need to send the address of the device and then the internal register address specific to the X-axis.

After the addressing sequences, the data streams are as many as they are sent until the data is completely sent and ends with a special stop condition.

1.3 Experimental procedure

Now we will build the circuit. This circuit consists of an LED that lights up when power is applied, a resistor to limit current and a power supply (the Rasp).

- Connect the Ground PIN from Raspberry in the breadboard blue rail (-). Raspeberry Pi models with 40 pins has 8 GPIO ground pins. You can connect with anyone. In this experiment we will use the PIN6 (Ground);
- Then connect the 3.3V pin in the red rail (+).
- Now let's conect the SCL and SDA wires. Connect them like as shown in the Picture ;
- Now push the sensors in the breadboard;
- And insert the jumper wires connecting the sensor in the bus, like as shown in the Picture .

The Figure **??** shows how the electric connection is made:

1.4 Connecting remotely

Through your local Pharo image, let's connect in the Pharo image by running on Raspberry, enable the auto-refresh feature of the inspector, and open the inspector. Run this code in your local playground:

```
remotePharo := TlpRemoteIDE connectTo: (TCPAddress ip: #[193 51 236
    212] port: 40423)
GTInspector enableStepRefresh.
remoteBoard := remotePharo evaluate: [ RpiBoardBRev1 current].
remoteBoard inspect.
```

1.5 Experimental code

In your inspect window (Inspector on a PotRemoteBoard), let's create the instances of the firt sensor, .