Interfacing with Sensors Practical

Introduction

Hopefully, you've read how to access the digital and analogue pins of the Arduino by using the simple example of blinking the LED. The LED is an example of a basic actuator i.e. it is controlling some action in the outside world.

Interfacing sensors and other actuators basically follows the same methodology as the one above. If it is a sensor, the pin must be setup as an input, if it is an actuator, the pin is now set up as an output.

Sensors give our microcontrollers insights into the physical world, they are sort of the eyes and ears of the microprocessor kind of the way your eyes and ears work with the brain.

Remember our discussion about interfacing?

We mentioned that interfacing is basically connecting or communication that happens between different ICs. We mentioned 3 examples of serial communication technologies UART, I2C and SPI.

We will not deal with any actuators (except the LED) but the logic is the same as with sensors.

For now, our sensors will be primitive and will not be using any of these communication technologies but the logic again remains the same.

Connecting Sensors with the Arduino

Many sensors have one output pin which is connected to the microcontroller and which the microcontroller uses to read and interpret the sensor's output. The output is usually a voltage between 0 and 5 v or something like that.

Given any sensors (and actuators for that matter), you can follow the simple and straightforward guide below to interface with the Arduino.

1. Search the internet for the sensor's data sheet or at least a detailed description of the sensor. Is the sensor analogue or digital? Take note of important things such as the working voltage, current draw, accuracy, operating temperature, etc. Basically anything you can find out about it is useful. Such information is valuable when you are designing your circuit e.g. the minimum voltage you need for it to work.
2. Find out how the sensor works and how you can use it. Know what is the output of the sensor and how your microcontroller will use it.
3. Finally, find out if there are any available libraries (which you are most definitely assured to find) to assist you in reading the sensor. There is no need for you to re-invent the wheel. If someone has abstracted the hassle of having to deal with the sensor outputs yourself, save yourself some time by downloading the library and using it. Libraries are important in any programming language.

Now let's do some practical stuff.

We'll interface at least one analogues sensor, one digital sensor and one actuator (in this case we'll just use the LED to make things easier).

Let's kick off and interface the Temperature and Humidity sensor (DHT11) to our Arduino.

The Link below gives us all we need to know to begin working on the DHT sensor

https://learn.adafruit.com/dht/overview

Now, let's see if we can be lucky enough to find a library!

Go to the Arduino IDE, click on Sketch, then Include Library, then Manage Libraries

Wait for the indexing to load. On the search pane, type DHT, and immediately you'll see a couple of libraries popping up. Choose the one from Adafruit and install it.

If you don't find a library there, don't panic ☺. What is listed there is in the official Arduino repositories. I sometimes find helpful library from such places as GitHub which are not listed there. Just do a simple internet search for the name of the sensor, you'll most likely find an Arduino library for it.

As if that was not simple enough, almost all these libraries come with examples. Just click on File Then Examples. Search for the name of the library you installed to find the example usage. From here on you can copy the example code in your own sketch and tweak it to do what you want. That's it.

Now let's look at an Analog Sensor. In our case we are dealing with the MQ2 Sensor which is a smoke detector.

Let's try to get a library like we did for DHT. I searched 'mq2', then 'MQ2'(sometimes writing it in caps makes a difference). Looks like this time we may not be that lucky. Okay, but we skipped a step earlier. Do you remember what it was?

We were supposed to search for a description on the sensor first and get all information about it.

The following links are sufficient for us to understand the MQ2 sensor

http://wiki.seeed.cc/Grove-Gas_Sensor-MQ2/

http://www.instructables.com/id/How-to-use-MQ2-Gas-Sensor-Arduino-Tutorial/

I am going to use the simple code provided by the sample code on the first link, but do read the rest of the document to know exactly how the sensor works and conditions to use it.

Note, that if we were rolling this sensor in production, it becomes very important to calibrate it. Here is how I would do it:

-Find out the gas that is in question, what gas are we monitoring the concentration of (The sensor should probably not be used to measure several gas concentrations at once)

-Find out the average temperature the sensor will be operating in

-Assuming I have another way of measuring the gas levels accurately, I would start with low gas concentrations and read the output of the sensor and work my way up (in small equal steps) to very high concentrations of that particular gas. Then I would draw the response graph and compare it with the actual values to determine how I will calculate the exact concentrations from the raw values read by the microcontroller. Any idea of how you would do it differently?

If for some reason we cannot calibrate the sensor ourselves, the datasheet should provide those graphs for different gases.

Let's move on. The usage is almost identical to that of the digital sensor (DHT 11) except that we are reading from an analogue pin and not using any library, we are using the values raw!

Assignment

You have learnt how to control led lights, read digital sensors, and read an analogue sensor. Now, I would want you to combine that knowledge to build a useful program that we can use to detect fire. Let's assume you want to determine if there is a fire by the temperature and the smoke concentrations in a room. Use what you have learnt to build a circuit that reads the temperature and gas concentration levels. If the temperature is above 27 (ideally this is too low to be a fire but we can achieve this temperature in a lab environment easily) and the smoke levels above the normal levels (since I don't see how you will generate enough smoke to simulate an actual case of fire, find out the normal levels and set the threshold such that it will always be passed with normal concentrations).

We are not done yet, when the concentrations are normal (our simulated normal), light the green (or white) LED continuously. When both thresholds are passed, blink the Red LED intermittently (on and off each second) so it might draw attention. Remember to use functions (at least have two) whenever you can. Good Luck!

Another thing, if you have a white LED instead of green, use it in place of the green one.

After you implement the above system (it's a whole system since it consists of both hardware and software), you would have designed an actual thing. While it's not yet an internet of things yet since you have not connected it to the internet, but we have done quite a lot. We have collected interpretable data, we now only need to add a communication device to send our data to the internet. I'll take you through adding communication modules to your things later.