SYSC3010 - Systems Development Lab
Lab 3 : GPIO

LAB OBJECTIVES

- Exercise the Pi General Purpose Input/Ouput, by adding simple circuits to the Pi
- Learn how to use an Arduino
- Learn how to communicate between a Pi and an Arduino over serial.

BRING YOUR TEAM'S KUMAN KIT

CAUTION: All hardware changes must be done with the power off

SUBMISSION

Each team only has one Kuman kit containing 1 Arduino controller. For this reason, your team shall work in pairs.

- In the first half of the lab, one pair will work on Part 1 (with the Pi) and the other pair will work on Part 2(with the Arduino)
- In the second half of the lab, the pairs will switch parts.

While you are in the lab, please demo the following portions to the TAs (for marks)

- Chapter 11 (blink_led.py), Chapter 13 (button_press_led.py), Activity 4.b, Activity 4.c and Activity 4.d

Additionally, each individual will submit their (paired) work via a CULearn assignment, proof of their work.

- Everyone must submit the actual program for each activity (Total: 5 files).
- For those who cannot finish within the allotted lab time, you must also submit photographs (taken from your phone) of any activity that you did not demo to the TA, For any activity with a Pi, your photo must show the inventory number of your Pi; for the Arduino, make sure your Kuman box is visible in your picture. (Maximum: 5 files)

REFERENCES

[1] Lee, Assam. *2019 Ultimate Guide to Raspberry P: Trips, Tricks and Hacks*. Packt Publishing 2019.

- (Lab 3) Chapter 9-14: Working with GPIO pins
  - Chapter 9.3 : Downloading code and Resources from GitHub

[2] Donat, Wolfram. *Learn Raspberry Pi Programming with Python: Learn to Program on the World's Most Popular Tiny Computer*. Apress 2018.

- (Lab 3) Chapter 13: The Raspberry Pi and the Arduino (Installing the Arduino IDE on the Pi)

[3] Warren, Gay. *Advanced Raspberry Pi: Raspbian Linux and GPIO Integration*. Apress 2018.

- Chapter 11: GPIO Hardware (Background Theory)
- Chapter 13: C Program GPIO (Background)

[4] Raspberry Pi and Arduino: What's the Difference and Which is Best for Your Project? Available at https://interestingengineering.com/raspberry-pi-and-arduino-whats-the-difference-and-which-is-best-for-your-project

PART 1 GPIO EXPERIMENTS ON THE PI

1 Background Read

You must begin by reading. Read [3], Chapter 11. You must read this chapter, before adding any circuits.

Some cautions from this book:

- The Pi's GPIO interface uses a weak CMOS 3.3 V interface. Circuits must be designed properly if using 5V TTL logic.
- The Pi's GPIO pins are susceptible to electrostatic discharge
- The pi's GPIO pins are weak drives, providing only 2 to 16 mAmps, from a total space current capacity of 50 mA.
- GPIO must be configured before use, because of the many models of Pis.
- All GPIO pins except the I2C pins are pulled high or low by an internal 50 kohm resistor, but they are weak and provide only default states when unconnected. So external resistors should be used when adding a circuit.

2 Activities

Create a personal Git repository for the files that you create in this part of the lab (Filenames are given below). The TAs will inspect these files as part of the lab. You will also submit a photo of each of your circuits.

You will follow along with the activities in [1], starting at Chapter 9.

Chapter 9: Working with GPIO (Getting resources from GitHub)

Chapter 10: Powering an LED

Chapter 11: Blinking an LED (blink_led.py)

Chapter 12: Detecting a Button Press (button_press.py)

Chapter 13: Using a push button switch to control a LED (button_press_led.py)

Chapter 14: Using a Passive Infrared Sensor (pir_sensor.py) [Optional, if you can find an IR sensor in your KUMAN kit]

If you're interested, or for your project, here's another idea for GPIO.

-

PART 2 GETTING STARTED WITH YOUR ARDUINO

1 Overview

This portion of the lab document is intended to provide directions on how to use Arduino, and provide some useful resources where you can find answers for your questions, or at least some of them. In addition, there is a list of activities to be done on this lab:

- LED blinking (section 4.b);
- LED blinking with variable time (section 4.c);
- Analog readings(section 4.d); and
- Use of timer interrupts (section 4.e).

2 Useful resources

- Arduino Software IDE - Information about how to use the Arduino Software Integrated Development Environment
- Getting Started - the main link where you can find information about software download, short tutorials, libraries, troubleshooting...
- But you can also look into tutorials, references, and playground, where you will find a lot of useful information, or find your answers for everything (or almost everything).
- Want to make good schematics and show it on your report, take a look at fritzing.

3 Prerequisites

1. Software
   (a) Install the Arduino Software IDE on your laptop, or on a Pi (See instructions in [2] Chapter 13). (Select the version that does not need admin privileges in case you are going to use the lab computers).
   (b) Library "*MsTimer2.h*" (to be installed in 4.e).
2. Hardware
   (a) 1 Arduino Uno board
   (b) 1 Variable resistor 10kΩ
   (c) wires

4 Lab Activities

4.a Procedure for ALL activities

1. Open the Arduino IDE.
2. Connect your Arduino board to a USB port of your computer.
3. Be sure that the software is configured for your board/connection.

   Go to: **Tools→ Board:**

   Select "**Arduino/Genuino Uno**"

   Then select the right COM port (Serial* [1]) by going to: **Tools→ Port**

   Select the correct COM Port. You can use "Device Manager" to identify the right port if you have more than one available.

4.b Activity - LED blinking (wow!)

Let's blink the on-board LED of the Arduino board and take a look at the code provided.

1. On Arduino IDE, open the blink example: **File→ Examples→ 01.Basics→ Blink**
2. Click on the "**upload**" button.

You should now see an LED on your board blinking once a second. Look at the code and identify these functions:

- pinMode()
- digitalWrite()
- delay()

What do you think about the functions *setup()* and *loop()*? If you don't know, look here.

[1] As most of the computers doesn't have an actual COM port, the Serial COM port is emulated through USB.

4.c Activity - LED blinking with time varying using Analog input

For this activity you are going to need your breadboard, wires and a variable resistor provided in your Kuman k4 kit!

1. First, let's take the components we need and put all together following the schematic on figures 1 and 2 [2].

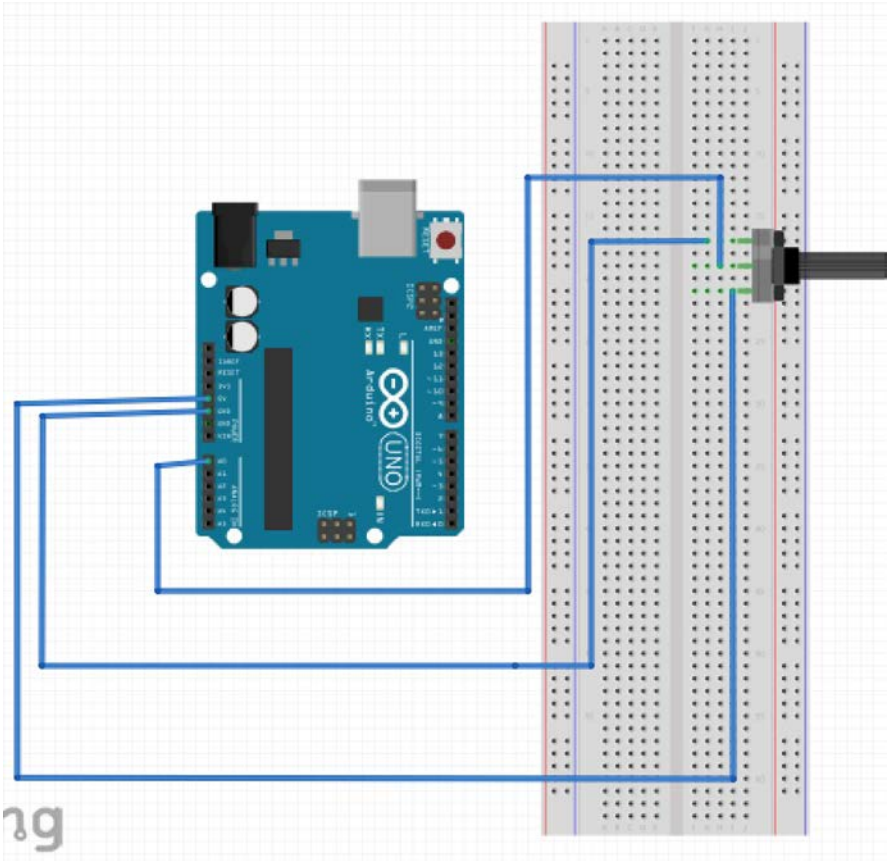

Figure 1: Breadboard representation of activity 4.c.

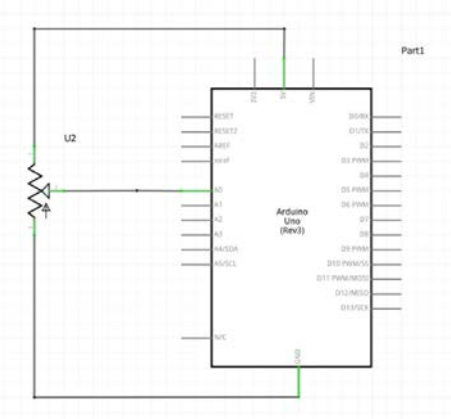

Figure 2: Schematic of activity 4.c.

[2] **Look at the diagram - connecting the wires at the wrong terminals might short-circuit your Arduino board.**

2. Edit the code from activity 4.b, following code in section 4.c.1
3. **Upload** the program into your board.
4. Turn the variable resistor in your breadboard and observe the time between blinks changing from zero (or close to that), to more than 1 second (recall ADC is 10 bits → 2^{10}).

Look at the code and identify these functions: *analogRead()*

4.c.1 Code

```
#define var_resistor A0

// Global Variables int var_resistor
_value ;

// the setup function runs once void setup () {
    //  initialize  digital  pin LED BUILTIN as an output .
    pinMode(LED BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever void loop () {
    // read the analog input from the variable resistor var_resistor _value = analogRead(
    var_resistor );

    // turn the LED on (HIGH is the voltage level ) digitalWrite (LED BUILTIN, HIGH);

    // wait for the amount of time read from variable resistor delay ( var_resistor _value );

    // turn the LED off by making the voltage low digitalWrite (LED BUILTIN, LOW);

    // wait for the amount of time read from variable resistor delay ( var_resistor _value );
}
```

4.d Activity - Serial communication + plot results

For this activity we will use the code from activity 4.c and write more code. Sometimes, it is good to actually see what is happening, which might not be an easy task without looking what is happening with the code. Serial communication with the Arduino IDE is an easy way to look at your analog readings. So let's see how...

1. Edit the code from activity 4.c, following the code on section 4.d.1.
2. **Upload** the program into your board.
3. Click on Tools → Serial Monitor.

   At the bottom right corner, select the right baudrate for serial communication [3].

   With this tool you can see what are the values that Arduino is sending through the serial port.
4. Once you know that you are receiving data from your serial port, **close** the Serial Monitor. You can turn your variable resistor and see these values changing.
5. Let's now see those valued plotted in a graph, so go to Tools → Serial Plotter.

   Since the serial communication is already configured from previous steps, there is no need to change configuration. Now you should see the graph being plotted in real time [4].

You should now see an LED on your board blinking once a second. Look at the code and identify these functions:

- Serial.begin()
- Serial.println()

Important Serial information

What else can I do with serial or other available functions?

[3] Recall that, from the code you uploaded to the board, the baudrate was configured to be 115200 bits/s.

[4] Since the time between analog readings will be varying because of the delays based on the values read, the plot will also be varying. To be a real time plot, it would be nice to realize both readings and serial communication apart from your code. That is why section 4.e is here

4.d.1 Code

```
#define var_resistor A0

// Global Variables int var_resistor_value ;

// the setup function runs once void setup () {
   // i n i t i a l i z e  d i g i t a l  pin LED BUILTIN as an output . pinMode(LED BUILTIN, OUTPUT);

    // begin the serial port with 115200 b i t s per second
    Serial . begin (115200);
}

// the loop function runs over and over again forever void loop () {
   // read the analog input from the variable resistor var_resistor_value = analogRead( var_resistor );

     // Send the value over serial COM port
    Serial . println ( var_resistor_value );

    // turn the LED on (HIGH is the voltage l e v e l ) digitalWrite (LED BUILTIN,
    HIGH);

    // wait for the amount of time read from variable resistor
    delay ( var_resistor_value );

    // turn the LED off by making the voltage low digitalWrite (LED BUILTIN,
    LOW);

    // wait for the amount of time read from variable delay ( var_resistor_value
    );
}                                                                    resistor
```

4.e Activity - Libraries/Interrupts

- Want to know about interrupts?
- Want to know more about timers?
- Want to know more about libraries?

In this section you will learn to install a library that is not part of the Arduino IDE by default. Also, you will use the code from previous examples, but now you want to blink the LED inside the *loop()* function, and read the values from your variable resistor every 20*ms*, which will allow you to visualize your graphs in real time, and with same period between readings.
So, let's start...

1. Go to "Sketch → Include Library → Manage Libraries".
2. At the "*Filter your search...*" box, type the name of the library we are looking for ("MsTimer2").
3. Click at the library you want, and click on *Install*. It might look as figure 3.

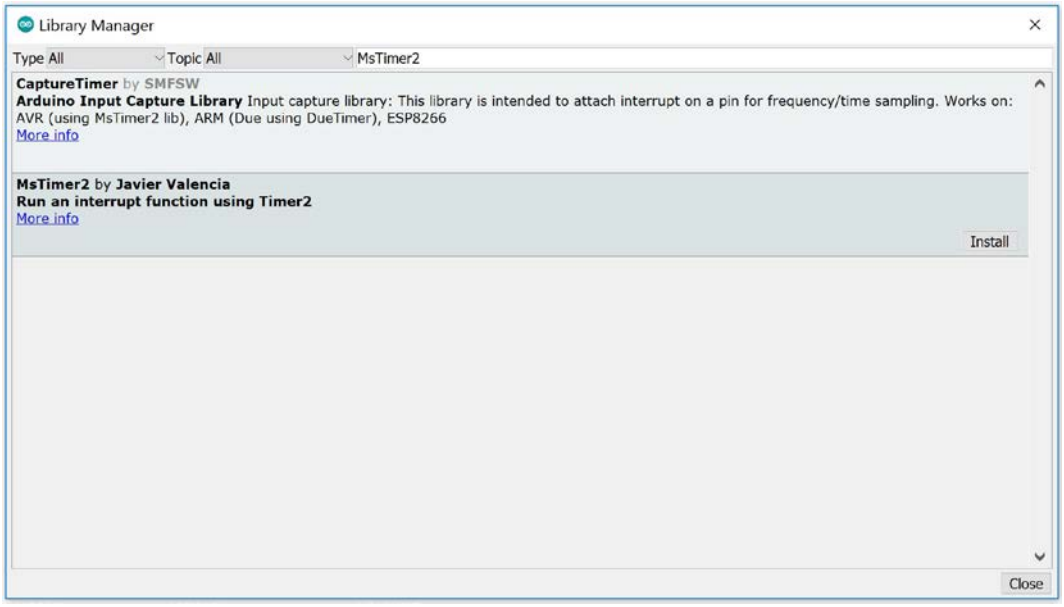

Figure 3: Example of how to install a library.

4. Use the code on section 4.e.1. Note that you need to include the library installed by adding the line "*#include <MsTimer2.h>*". Analyze and understand this code...
5. **Upload** the program into your board.
6. Open your serial Plotter.
7. Turn the variable resistor and observe the differences from the code implemented on section 4.d.

4.e.1 Code

```
#include <MsTimer2 .h>
#define var _resistor A0

// Global Variables int var
_resistor _value ;

//   Function   prototypes   void
readVariableResistor ();

// the setup function runs once void setup
() {
  // i n i t i a l i z e     d i g i t a l     pin LED BUILTIN as an
  output . pinMode(LED BUILTIN, OUTPUT);

  // begin the   serial      port  with  115200      b i t s     per
     second Serial . begin (115200);

  //  Configure  timer2  interrupt  to  happen  every  20 ms
  // calling     the   function    readVariableResistor
  MsTimer2 : : set (20 , readVariableResistor );

  //  Start  the  timer2  interrupt
  MsTimer2 : : start ();
}

// the loop function runs over and over again forever void loop ()
{
  // turn the LED on (HIGH is the voltage l e v e l ) digitalWrite
  (LED BUILTIN, HIGH);

  // wait for the amount of time read from variable resistor delay (
  var _resistor _value );

  // turn the LED off by making the voltage low digitalWrite (LED
  BUILTIN, LOW);

  // wait for the amount of time read from variable resistor delay (
  var _resistor _value );
}
```

```
void readVariableResistor (){
  // read the analog input from the variable resistor var _
  resistor _value = analogRead( var _resistor );

  // Send  the  value  over   serial COM port
  Serial . println ( var _resistor _value );
}
```