Baylor University

Department of

Electrical and Computer Engineering

BME/ELC 4372

Bioinstrumentation Laboratories

Keith Schubert
Professor
Department of Electrical and Computer Engineering
Baylor University

Contents

Lab 1

Raspberry Pi

Over the course of this semester we will be building a variety of medical and biological sensors. We will be using the Raspberry Pi as our microcomputer to control them, because of its ease of use, large number of IO pins, and tons of example code to build on. In this lab we will be introducing the Raspberry Pi and how to use it. I am going to try to do most things in Python, due to its simplicity and extensibility, but I can't guarantee we won't have to do a little programming in another language.

1.1 Network

Your Pi either has a usb WiFi connector or built in WiFi. In both cases you need to log into the network using the network gui. Select it from the top right of the Pi's desktop, select the network then enter the login information.

1.2 GIT

One thing I really want to bring to your attention is the use of **git**. Git is a version control system (VCS), that was designed by Linus Torvalds to handle the development of Linux. I will be maintaining a git repo at github, which means you will be able to clone it and do a pull any time you want to update it. You do not have to use git, it just saves time and is a good skill to know for industry. Open up a terminal (upper left of the desktop icon is black and blue with a white `>_`). Type the following:

```
ls
```

If you don't see a directory named `code` then you will need to make it by typing:

```
mkdir code
```

Now you need to change to the code directory and get the git repo with starter code.

```
cd code
```

```
git clone https://github.com/BaylorBioMedicalEngineering/BME-4372-BioInstrumentation-labs.
git1
```

```
cd labs
```

```
ls
```

1.2.1 Some commands for reference

To update from the main repository, just do a pull

```
git pull
```

More sample code can be found by:

```
cd  /code

git clone http://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git

cd Adafruit-Raspberry-Pi-Python-Code
```

1.3 Das Blinken LED

- Raspberry Pi (with keyboard, mouse, monitor, cobbler, and breadboard)
- 220Ω - 330Ω Resistor
- LED

First you need to assemble the circuit. Hook up a wire from a gpio pin to the anode (long leg) of the diode, then connect the resistor to ground. The resistor limits the current flow. The Pi will output 3.3V and the diode will cause about a .7V drop resulting in a 2.6V drop left over. The remaining 2.6V flowing through around 220Ω to 330Ω will result in around 10mV, which is enough to light the LED but not cause problems sinking or sourcing the current for the Pi. Generally be careful with more than 20mV - 30mV for an IC.

Boot the Pi by plugging it in. When the desktop appears, launch a terminal either from the menu or the terminal button. Type the following line to edit the code to turn on and off with a timer.

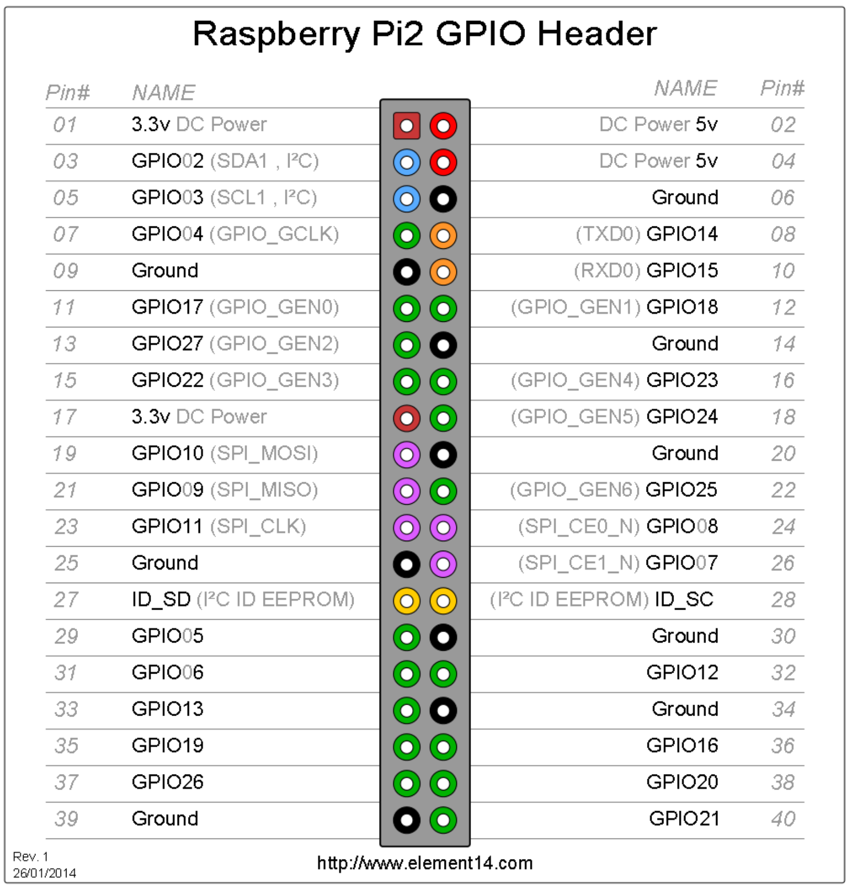
```
sudo nano das_blinken_light.py
```

Note that nano is a small editor, hence the cute name. You are welcome to use any editor you like. You should see something that looks like Code 1.1. The only thing you have to edit is the gpio pin number to match the one you used, see Figure 1.1.

Listing 1.1: Blink the Light.

```python
# das_blinken_light.py

import RPi.GPIO as GPIO
import time

ledPin = 23

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.output(ledPin, GPIO.LOW)
```

Figure 1.1: Raspberry Pi 2 General Purpose Input Output (GPIO) pinout.



```
try:
  while True:
    GPIO.output(ledPin,GPIO.HIGH)
    time.sleep(1)
    GPIO.output(ledPin,GPIO.LOW)
    time.sleep(1)
except KeyboardInterrupt:
  GPIO.cleanup()
```

To run the code you need to type the following in the terminal window.

```
sudo python das_blinken_light.py
```

You need sudo to give enough permission to access the gpio pins. The LED should blink on for a sec and off for a sec.

1.4 Das Blinken LED II

- all the previous parts
- switch

Hook up with switch or button to ground. The Broadcom chip that runs the gpio for the Pi has the ability to connect a pullup or pulldown resistor to an input. We will thus use a pullup resistor, and the button will pull it down to ground when pressed.

We will now write code to read input and blink led if the button isn't pressed. Type in:

```
sudo nano das_blinken_light_II.py
```

You can also refer to the GitHub repository, if need be.

Listing 1.2: Blink the Light while the button is not pressed.

```
# das_blinken_light_II.py

import RPi.GPIO as GPIO
import time

ledPin = 4
butPin = 22

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(butPin,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(ledPin, GPIO.LOW)

try:
  while True:
    GPIO.output(ledPin,GPIO.input(butPin))
    time.sleep(.1)
    GPIO.output(ledPin,GPIO.LOW)
```

```
    time.sleep(.1)
except KeyboardInterrupt:
  GPIO.cleanup()
```

```
sudo python das_blinken_light_II.py
```

1.5 Das Blinken LED III

One last thing for today is to dim the led. The Raspberry Pi does not have an A/D converter, so we will just deal with two levels based on the button. We also only have one pulse width modulated output on Broadcom (BCM) pin 18, which is Board pin 12. Lack of A/D converters is one of several reasons it is not a replacement for a micro-controller, the main one being it doesn't have a real time operating system (RTOS) and lacks the necessary timers, like watchdog timers, to build one. If you ever need a micro-controller then use an Ardino, MSP430, or similar. The Pi has a bunch of advantages too in standard tools, quad core, and so on. We care more about the later for labs and testing. If you ever build patient used tools, get a micro-controller as you need the RTOS.

Getting back to the point, we want to have the LED on all the time now, and we will dim it when the button is pushed. To handle the diming, we will use a pulse width modulator. We will set the frequency to 50Hz, which is a reasonable refresh rate, and will use the duty cycle (percent of the wave that is high) to control the brightness. This is a simple and standard way to handle this. Type in:

```
sudo nano das_blinken_light_III.py
```

You can also refer to the GitHub repository, if need be.

Listing 1.3: Dim the Light while the button is pressed.

```
# das_blinken_light_III.py

import RPi.GPIO as GPIO
import time

ledPin = 18
butPin = 22

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(butPin,GPIO.IN, pull_up_down=GPIO.PUD_UP)
pwm = GPIO.PWM(ledPin, 50)
pwm.start(100)

try:
  while True:
    if GPIO.input(butPin):
      pwm.ChangeDutyCycle(100)
    else:
      pwm.ChangeDutyCycle(25)
except KeyboardInterrupt:
  pwm.stop()
  GPIO.cleanup()
```

```
sudo python das_blinken_light_III.py
```

Lab 2

Electromyography

2.1 Biopotentials

Biopotentials are formed by ion concentration differences inside and outside a cell. Membranes and specialized pumps in the membrane regulate and adjust the concentrations in response to external and internal stimulation, permitting the generation and propagation of biosignals. Measuring the electrical potential in muscles is called electromyography or EMG.

Generally, doctors use needle electrodes, so the skin only needs to be wiped with alcohol to prevent infection, but we will not be using needle electrodes for safety and legal reasons (Texas Court of Appeals, Third District, at Austin, Cause No. 03-10-673-CV. April 5, 2012)++.

2.2 Setup

First, we will get the Arduino IDE. You we need to make sure we have the latest version of our software. Open a terminal window and enter the following commands:

```
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install arduino
```

The ArduBerry is a shield that allows Raspbery Pi's to use Arduino shields. Type the following commands:

```
cd code

git clone https://github.com/DexterInd/ArduBerry.git
```

Open a web browser and go to `http://www.dexterindustries.com/Ardubecrry/getting-started/` and follow the instructions. You will download and install the drivers and run a simple test script that verifies everything is working. The brief summary is below (the website has pretty pictures to accompany this):

1. Stack ArduBerry on Raspberry Pi.
2. Go to the scripts directory in the ArduBerry repo : `cd ArduBerry/script`

3. Make the install script executable: `sudo chmod +x install.sh` then run it as root: `sudo ./install.sh`

   and follow prompts, pressing `enter` and `y` as needed.

4. The system should automatically reboot.
5. Open the Arduino IDE from the system menu.
6. From the **Tools** menu, select the **Programmer** sub-menu, then select **RaspberryPi GPIO**.
7. Load **blink** or another sample sketch, and press **CTRL+Shift+U** to upload (or select **Upload using Programmer** from the File menu)
8. Verify that the LED blinks (if running blink)

2.2.1 Olimex EKG/EMG Shield

You should not have to perform any installs for this. The shield is static sensitive so be careful, and the pins are often bent so be more careful. Disconnect power. Insert onto the Arduberry. Connect power. From Pi launch Ardino editor, and set the target device to an uno and the programmer to GPIO. **Load arduberry emg sketch**, then **program**.

Now open a command window and cd to your EMG directory. Run EMG.py. This is a simple test program, that should give you 8 numbers (1-4 interspersed with something around 300). If you get this all is well.

Now connect a volunteer to the ekg leads. L goes on the left arm, R goes on the right arm, and D goes on the right leg. They need to be symmetrically placed, i.e. all at wrist/ankle or elbow/knee, etc.

Run either the fixed (takes 2k samples then plots and holds) or plot (for dynamic plots).

2.3 General Advice

Gel electrodes provide a good measure but a few basic precautions should be followed to get the best signals.

- Remove oils from your skin with soap and water to improve the signal.
- Don't apply lotions or creams.
- Make sure you are hydrated (dry skin doesn't conduct as well).
- Try to use smooth skin with minimal hair (callouses and hair make it more difficult to get a good signal).
- Body fat reduces the signal, so try to find areas where the muscle is as close to the surface as possible.

2.4 Test

Place ground on bicep. What happens if you use the right leg?

Place other two on the same forearm, locating muscles for two different fingers on the same hand by extending and contracting individually and noticing muscle flexure. What happens if you use the opposite arm?

Record signals for motion and identify finger flexed by chart only. Explain.
How does the amount of exerted force relate to the signal?

Lab 3

Electro-Kardio-Gram

Lab 4

One Wire Thermometer

4.1 Add One Wire Support

First you need to edit the boot configuration file to add one wire support.

```
sudo nano /boot/config.txt
```

Scroll to the bottom (use the down arrow), then type **dtoverlay=w1-gpio**. Save by typing **ctrl-o** then exit with **ctrl-x**.

Now reboot to make your changes active.

```
sudo reboot
```

4.2 Test It

```
sudo modprobe w1-gpio
```

```
sudo modprobe w1-therm
```

```
cd /sys/bus/w1/devices
```

```
ls
```

Note that the next directory has a really long name, and incorporates the device number so it is not constant on all systems. This allows multiple to be connected.

```
cd 28*
```

```
cat w1_slave
```

4.3 Code It

Listing 4.1: Read from a One Wire Thermometer.

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

device_folder=glob.glob('/sys/bus/w1/devices/28*')[0]
device_file=device_folder+'/w1_slave'

def read_w1_file():
  f=open(device_file,'r')
  lines=f.readlines()
  f.close()
  return lines

def read_temp():
  lines=read_w1_file()
  while lines[0].strip()[-3:] != 'YES':
    time.sleep(0.2)
    lines=read_w1_file()
  temp_loc=lines[1].find('t=')
  if temp_loc != -1:
    temp_string=lines[1][temp_loc+2:]
    temp_c=float(temp_string)/1000.0
    temp_f=temp_c*9.0/5.0+32.0
    return temp_c, temp_f

while True:
  print(read_temp())
  time.sleep(1)
```

Lab 5

Pulse Oximeter

5.1 Background

Hemoglobin is a protein in red blood cells that reacts to oxygen forming oxyhemoglobin (HbO_2). When not oxygenated hemoglobin is referred to as deoxyhemoglobin (Hb). Pulse oximetry is a non-invasive way to measure the amount of oxygen dissolved in the blood, which is called the oxygen saturation (SpO_2). Oxygen saturation is measured by detecting Hb and HbO_2, using their absorption sectra at two different frequencies (typically red around 660nm and infrared around 840nm to 940nm). These values were selected because Hb has higher absorption of red light and HbO_2 has higher absorption of infrared, see figure 5.1.

We are going to use two LEDs, one red one infrared, a photodiode, and a simple filter/amplifier to measure the absorbed light at two frequencies. The data that produced the graph came from the tabulated molar extinction coefficient, e, for hemoglobin in water compiled by Scott Prahl using data from

- W. B. Gratzer, Med. Res. Council Labs, Holly Hill, London

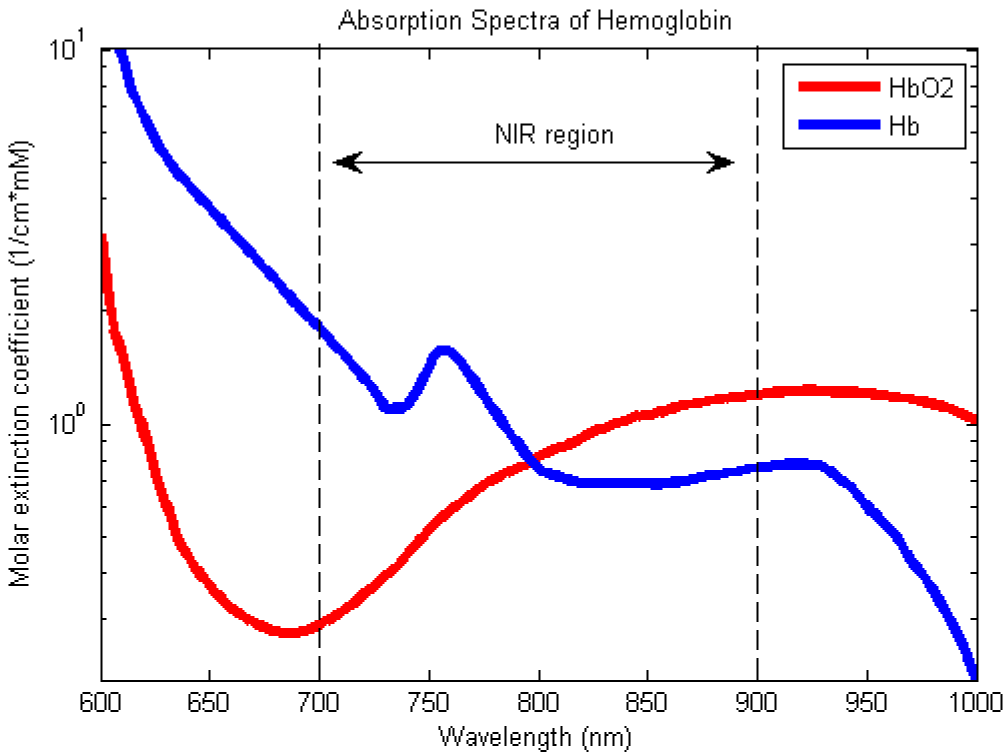
- N. Kollias, Wellman Laboratories, Harvard Medical School, Boston

$\lambda[nm]$	$HbO_2[cm^{-1}/M]$	$Hb[cm^{-1}/M]$
660	319.6	3226.56
830	974	693.04
840	1022	692.36
930	1222	763.84
940	1214	693.44

To get absorption, you multiply the molar extinction coefficient times the molar concentration times the pathlength and divided by the molecular weight of hemoglobin. Since we are comparing two absorption measurements at different frequencies, the molecular weight of hemoglobin cancels and can be ignored. Similarly, if we put both light sources (red and infrared) equally distant through your body to the photodiode, then the pathlength will also cancel and can be ignored.

Light is absorbed by tissue, venous blood, non-pulsatile arterial blood, and pulsatile arterial blood. The first three are constant and will be measured as the DC component of the measurements. The final one, pulsatile arterial blood, will be the AC component, and will also allow us to get heart rate.

Figure 5.1: Absorption spectra of oxyhemoglobin and deoxyhemoglobin. *Image by Adrian Curtin used under Creative Commons Attribution-Share Alike 3.0 Unported License.*



5.2 First Method

By taking the ratio of oxygenated arterial blood in the pulsatile (AC) over the other (DC) portion of the signal at two frequencies we can calculate the absorption ratio (AR)[1]

$$AR = \frac{\frac{AC_{red}}{DC_{red}}}{\frac{AC_{infrared}}{DC_{infrared}}} \tag{5.1}$$

We can then compare this number to previously measured values that were tested in a different manner. This is only approximate because of differences in blood volume, perfusion, etc. as well as movement and misplacement effect measurements. Even so we can make a simple table of values and interpolate to get other values:

AR	SpO_2
0.4	100%
1.0	85%
3.4	0%

[1] SpO_2 can also be calculated by ratio of the logarithms of the AC components at the two frequencies. Multiply by 100 to get percent.

Lab 6

Op Amps

6.1 MCP 3008

The MCP 3008 is an 8 channel, 10 bit A/D converter that is bus addressable. Communication is through a SPI bus. Our Raspberry Pi has hardware support[1]. There are four code sequences I have provided:

1. mcp3008bitbang.py - slow software SPI bus. Don't use, this is only for reference if you don't have a system with hardware support.

2. mcp3008hw.py - fast hardware SPI bus. outputs values on command line to 3 decimal places. Good for getting precise values, but bad for lots of values.

3. mcp3008plot.py - hardware SPI bus that plots the results on a graph. Tends to be slow because it has to sample and plot.

4. mcp3008plot-thread.py - multi-threaded hardware SPI bus that plots the result on a graph. Yes I was having fun... This has three threads, one that reads, one that outputs a square wave, and one that plots. Syntax for the plot commands follows MatLab standards. Fast and oh so much fun.

These, of course, require library support so we have written an install shell script. You will need to navigate to where it is located, change its permissions so it is executable, then run it. From a terminal window type:

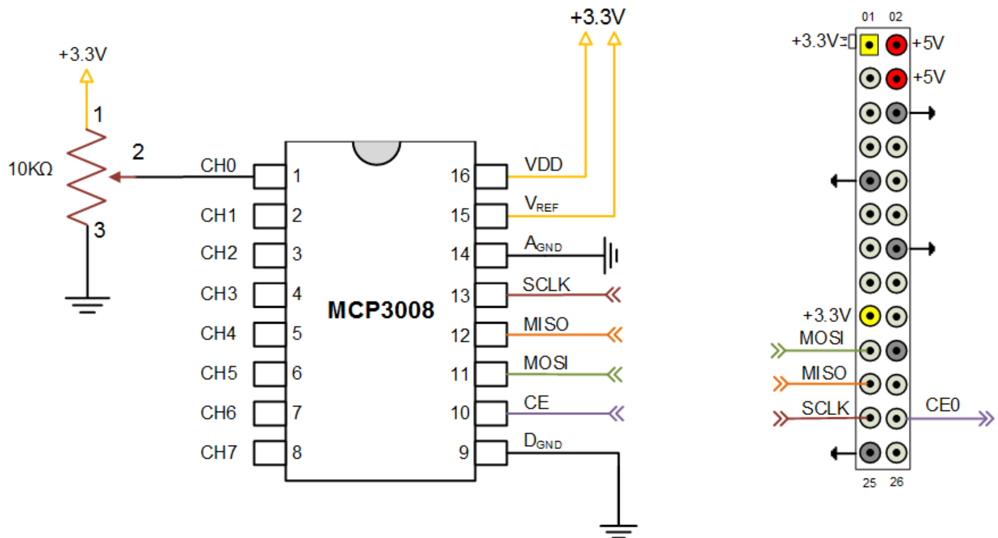
```
cd  /code/labs/labs

chmod 755 install_python_libs.sh

./install_python_libs.sh
```

The mcp3008 has an analog power reference(Vref) and ground and a digital power (Vdd) and ground. Vdd must always be connected to 3v3 (3.3V) and its ground (pin 9) to ground. Vref and analog ground define the voltage range to compare. Often this is the same, but it doesn't have to be. For instance, connecting to 5v gives a bigger swing. Remember that the chip only has 10 bits of precistion (just over 1000 divisions) and thus the precision is:

$$\frac{Vref - Agnd}{2^{10}}$$

[1]If you had a controller that didn't have hardware support you would have to implement it in software, which is very slow and called bit banging. I have included a bit banging code for your reference in the code part of lab 3.

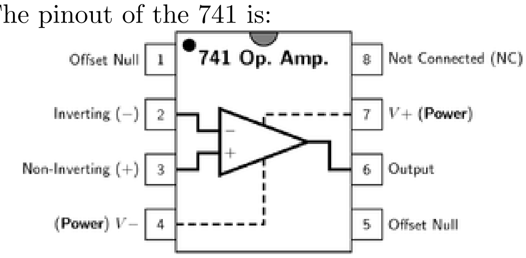
Thus the larger the reference swing, the less precise the measurement. If you want to measure something small, you should have a small reference! The basic hookup is thus



Note the voltage divider on the left is only for testing and does not need to be done each time. You will build the divider the first time to test your code and setup. The 8 pins on the left are now available for measuring your circuit.

6.2 Follower

We will be using a general purpose 741 Op Amp. We will connect the positive rail to 3v3 and the negative rail to ground. Hook the output to the negative input and the signal to follow goes on the positive input. The pinout of the 741 is:



6.3 Inverting Amplifier

Note we thus can't get negative voltages, but we have no choice since we don't have a negative voltage. We can make a new reference ground by a voltage divider between 5v and ground with identical resistors (say around 1k each) or a potentiometer around a few k[2], so the center will be 2v5. Hook this to the positive input, and the rails should be connected to 5v and ground. Hook the Vref on the MCP3008 to 5v. Now make another potentiometer voltage divider, this time running from the op amp's output to gpio 4, with the central tap going to the op amp's negative (inverting) input.

[2]hook 5v to one outer pin and ground to the other, the center tap is the reference

Lab 7

Electro-Encephlogram