ME 348E/392Q: Advanced Mechatronics

Laboratory 1: Introduction to Microprocessors

Submit one report in Canvas as a team. Include the names of your partners. Be sure to submit your completed worksheet (a scan) and any required plots, as well as attachments with your code in a single file (.pdf format). The due date is given on the Canvas assignment.

Overview
The purpose of this lab is to work through a series of tutorials to use both a Raspberry Pi and an Arduino-Like microcontroller to make a blinking circuit with a light emitting diodes (LEDs). These tutorials will give you a basic understanding of how microprocessors work and the differences between a microprocessor and a microcontroller. This lab also serves to help you set up the basic computing environment that you'll be using for the course project.

Components Required: You will need a Raspberry Pi, an SD card and SD/USB reader, a micro HDMI cable, a 220Ω resistor (or similar magnitude), an LED, and potentiometer(s) from your project box, your parts kit, or the instructor.

I. Pre-Lab Assignment: (DUE BEFORE FIRST LAB)

A. Watch this video on how to navigate the Raspberry Pi linux file structure. `https://www.youtube.com/watch?v=KQlF7IfgZ28&ab_channel=PaulMcWhorter`

B. Install the Raspberry Pi imager on your computer: `https://www.raspberrypi.com/software/`

C. Install the Arduino IDE on your computer if you don't already have it: `https://www.arduino.cc/en/software`

D. Review the Arduino introduction page: `https://www.arduino.cc/en/Guide/Introduction`

II. Flash the Raspberry Pi and Set Up RPi Environment

A. Open the Raspberry Pi imager with the SD card connected. Select the appropriate board (RPi 4B), the default operating system, and the SD card to use for flash. Click the gear icon for settings and make sure to enable SSH and select a username and password for your team (Team X and easy password for all to remember). Write to the card. This may take some time.

B. Plug the SD card into the slot at the bottom of your RPi. Also plug in a keyboard, mouse, and monitor. You can use the ones associated with the computer on your desk but please plug them back in for the next class. Finally, connect the RPi to power using the power adapter.

C. The RPi should now give you a welcome screen to select a location, timezone (use center), and get onto the internet (select UT guest). Skip the software update and click to restart. If it does not prompt these, you may have to set them yourself.

D. Once the RPi opens again, click the terminal icon at the top and type in: `sudo date -s 'YYYY-MM-DD HH:MM:SS'` command, where you replace the time variables to the current time. This can be found here: `https://time.is/CT`.

E. Now update the RPi with the following two commands: `sudo apt-get update`, (which might not have anything to do), and `sudo apt-get upgrade`. You'll need to confirm this last command with a 'y'. If these commands return errors, you may need to use a phone hotspot / alternative WiFi.

F. Install Visual Studio Code by typing `sudo apt install code` into the command window. After installation is finished, confirm you have python installed by typing `python3 --version` into the command window. You should have a Visual Studio as a programming menu option in the Raspberry Pi main menu (raspberry icon). Open that application, navigate to the extensions tab (icon with 4 cubes to the left of the screen) and install pylance. We will use this in a future lab.

G. Finally, we will install the git repo for this class.

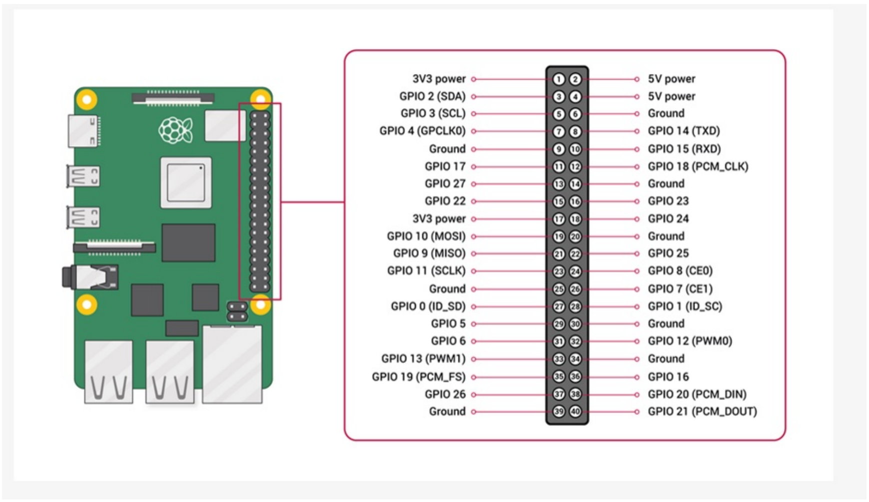
   i. Run `sudo apt-get install git-all` in the command window. Run `git version` to verify the installation worked.

   ii. Create a folder on the RPi desktop called Lab. Navigate to it using the command `cd Desktop/Lab`. Run `git init` to create a git folder inside this director (it's hidden). Now, in a web browser, create a github account if don't already have one and find the repository on Dr. Majewicz Fey's AnnMFey git hub titled `ME348_AdvMech_Lab1` and copy the URL.

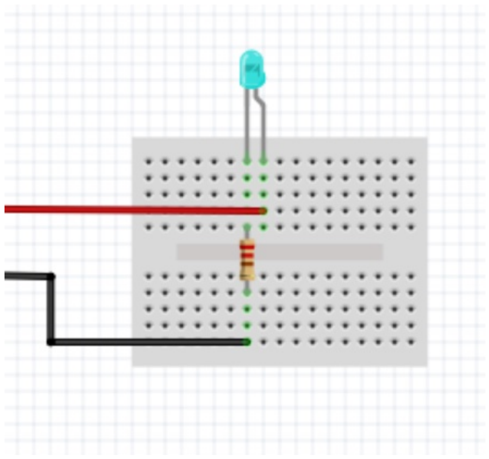
   iii. Back in the command line window, make sure you are still in the lab folder you just created and type `git clone`, pasting the URL. Note, to paste you need to right click to find the paste option, CTRL-V won't work. You should now have a folder titled `ME348_AdvMech_Lab1` in this folder.

III. Basic LED Circuit using Raspberry PI

A. Plug a 220Ω resistor (or one of similar magnitude) into your breadboard, in series with an LED. Make sure it works by giving the circuit power using the 5V power and ground pins on your RPi. The pin out diagram below will help you. Jumper wires will be available in the lab.


3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

B. *If you need more detailed guidance, see this reference. Note we aren't using a push button here, not yet.* ***See link for image.***



C. Now, open the `ME348_AdvMech_Lab1` python code and figure out what it does. What will this code do? How will you need to change your circuit wiring?

D. Back in the command window, making sure you are still in the correct directory, type `python Python_LED_blocking.py`. If all goes well, your LED should start blinking. Show the TA and use 'CTRL C' to kill the application.

IV. A More Interesting Basic LED Circuit

A. Modify your circuit so that you can vary the intensity of an LED using a potentiometer and also add a pushbutton to control turning on and off the LED. Draw your circuit below and show the TA or instructor that you have it working. For this you can go back to using just the 5V or 3.3V output on the Raspberry Pi.

B. Disconnect your circuit from the Raspberry Pi for the next step.

V. Using A-Star Controller to Blink an LED

A. Next, we are going to switch to the A-Star controller to control our circuit but will use Arduino to program it. Install the Arduino IED by navigating back to your home directory (../ takes you back one level) and type `sudo apt install arduino`.

B. Attach the A-Star controller by plugging it into the RPi's pin array, such that it is directly above the RPi board. Additionally, use the 6 inch USB-A to Micro-USB cord to connect the two boards.

C. The A-star Controller User Guide is a useful resource with information on downloading drivers and pin maps (`https://www.pololu.com/docs/0J66`). Specifically, Section 4.2 will describe how to get the right drivers in your boards manager. Make sure you can run blink with the on-board LED before moving on. If you are having difficulty uploading code to the board, you may need to reboot it by pressing the reset button twice while uploading the code. Change the frequency of the LED blinking in the code to prove it works (`https://www.pololu.com/docs/0J66/4.2`).

D. In the Arduino environment, click the File button on the toolbar and look in the Examples folder. Select the "Blink" example (in 0.1 Basics). Hit the check mark button to compile and the arrow button to upload the code onto the LaunchPad. Note: if you have errors you may need to install Java, the Java SDK, and update the boards and libraries on your computer.

E. Next, relaunch Arduino and try the "Blink without Delay" example in 0.2 Digital tab.

   i. Notice that this example has the same functionality as the "Blink" Example. What is fundamentally different about these two examples? For each code, describe some reasons why you think someone would chose one method over the other.

VI. Non-Blocking LED Circuit using A-Star Controller

VI. Non-Blocking LED Circuit using A-Star Controller Now we will integrate components of the circuit you built in part 4 of the lab with the A-Star controller.

A. Save the "Blink" example as a new file and change the name to "LEDBlinkingSpeed". Your objective is to change the code and your circuitry so that the LED in your breadboard blinks at a speed defined by the potentiometer value, but only after two button presses! *Hint: you need an analog input and to keep track of button states. Use serial.print() or serial.println() to debug.*

i. See these links for a similar examples:
`https://docs.arduino.cc/built-in-examples/analog/AnalogInput/`, `https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/`, `https://docs.arduino.cc/built-in-examples/analog/AnalogInOutSerial/` and look for pin outs here: `https://www.pololu.com/docs/0J66/3.9`.
The pin assignments page will help you figure out what pins are available for you to use `https://www.pololu.com/docs/0J66/3.10`

ii. Draw your circuit below or take a picture of it. Make sure to indicate what pins are used for what purpose. You will also need to save your code and include it with the worksheet submission (saved into one PDF file).

iii. Show your working system to the TAs or Instructor. Congrats on writing your first state machine (at least for this class)!
3 Commit this code back to git and upload this worksheet with your code appended to Canvas.

VII. Post-lab Reflection

A. It wouldn't have been possible to use the potentiometer as an input like you did in the last section if you only used the Raspberry Pi. Tell me why not. If all you had was the Raspberry Pi and not something like the A-Star controller, what external piece of circuitry would you need? Is that easy to integrate? Prove it by listing an online resource you can use next time.