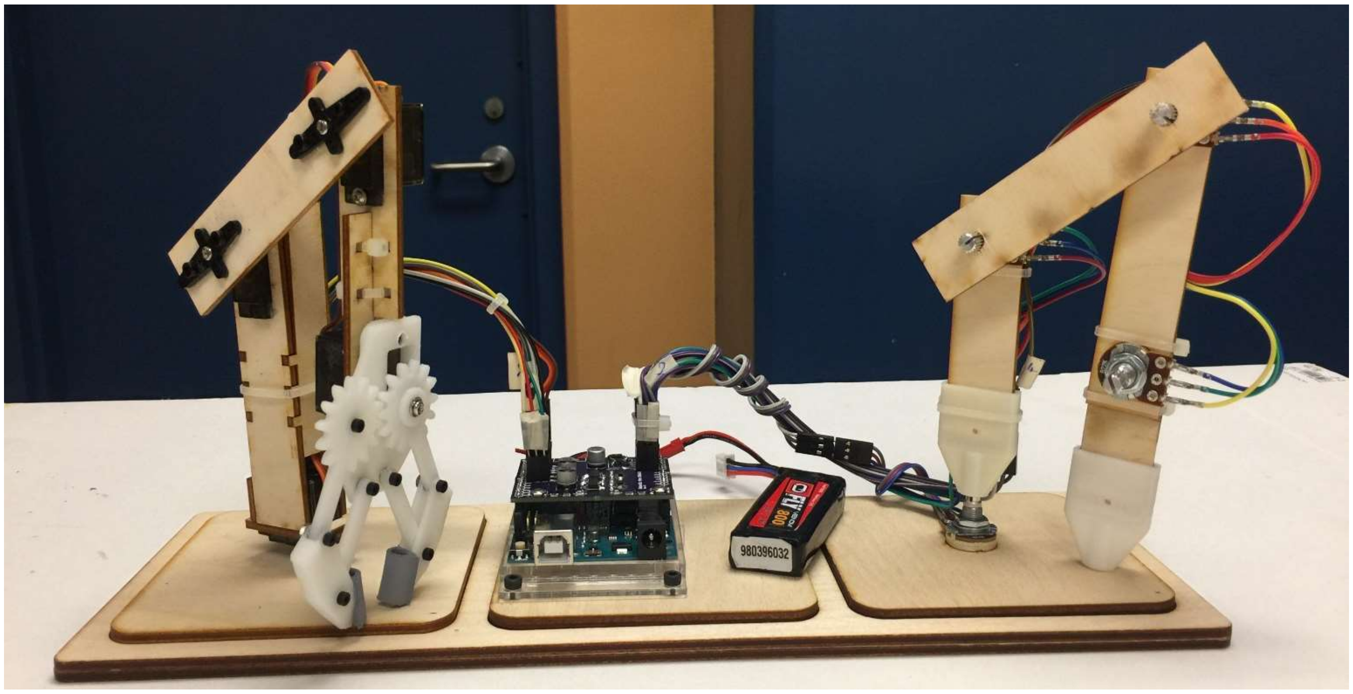
Mini Robotic Arm

Automation



LEVEL: ADVANCED

DEVELOPERS: Buu Truong, Jordi Medellin

ADVISOR: Professor Truong Nguyen, ECE

REFERENCES: Micro Servo Robot by Pinaut

PROJECT SEQUENCE: Mini Robotic Arm

LAST UPDATED: 10/13/2017

REQUIRED DOWNLOADS / INSTALLATION

1. Latest version of Arduino: https://www.arduino.cc/en/Main/Software
2. Latest version of EAGLE: https://www.autodesk.com/products/eagle/free-download

A NOTE FROM THE DEVELOPERS

Dear Student,

The ECE 196 course and Project in a Box team would like to give credit to Pinaut for making his Micro Servo Robot, which inspired us to create a similar project as one of the choice of this class' projects. We hope, through the process of building this robotic arm, you will gain a fundamental understanding of how a robotic arm function.

Sincerely,
P.I.B Team

INTRODUCTION

This project uses a robotic arm with a small gripper from your previous project, where a user motions the controlling arm, and the mechanical arm mimics the movement of the user controlling arm. Therefore, the user can perform precision tasks with the robotic arm. In part 3 of this robotic arm sequence, we aim to create a custom PCB board for the robotic arm, and, most importantly, we will program the arm to have the ability to learn a set of motion by imitating the controlling arm and repeat such motion on its own.

OVERALL LEARNING OBJECTIVES

The main goal of this project is to develop an algorithm that gives the robotic arm the ability to autonomously perform a set of motions, which is recorded during the time following the movement of the controlling arm. The algorithm requires us to write the motion of the controlling arm to the servo arm, and, at the same time, record such motion onto the Arduino dynamic memory. We can achieve this by simulating multithreading programing technique (proto-threading) on the Arduino. In general, the arm will have two extra buttons: a play button, and record button. Upon pressing the 'record' button, the arm will start recording the motion of the controlling arm until 'play' button is pressed, or the memory storage is full. Upon pressing the 'play' button, the arm will repeat the recorded motion in an endless loop.

OVERVIEW

CHALLENGE #1: PCB DESIGN WITH EAGLE

CHALLENGE #2: SMD/THD SOLDERING OF PCB BOARD

CHALLENGE #3: PROGRAM AUTOMATION MODE

CHALLENGE #4: MEMORY MANAGEMENT

REQUIRED PROJECT PARTS (ALL)

Product Name	Vendor URL	Quantity	Notes
POWER_JACK-BARREL_ID2.1MM-OD5.5MM	http://www.memoryprotectiondevices.com/datasheets/EJ508A-datasheet.pdf	1	
CAP_CERAMIC_SMD-805_0.1UF-50V-10%	http://psearch.en.murata.com/capacitor/product/GRM21BR71H104KA01%23.pdf	2	Ceramic Capacitors
CAP_CERAMIC_SMD-805_0.22UF-50V-10%	http://psearch.en.murata.com/capacitor/product/GRM21BR71H224KA01%23.pdf	2	Ceramic Capacitors
VREG_LM340_LDO_1.5A-5V	http://www.ti.com/lit/ds/symlink/lm340.pdf	2	
CAP_POLARIZED_ALUM-ELECTROLYTIC_SMD_100UF-16V-20%	http://industrial.panasonic.com/www-cgi/jvcr13pz.cgi?E+PZ+3+ABA0120+EEEFK1C101P+7+WW	2	Polarized Capacitors
RESISTORSMD-0805_1/8W_10K_5%		3	Resistors
DIODE_GEN-PURPSMD_GP_1A-50V-SOD123F	http://www.comchiptech.com/cms/UserFiles/CGRKM4001-	2	

	HF%20Thru.%20CGRKM4007-HF%20RevG.pdf		
RESISTORSMD-0805_1/8W_1K_5%	http://www.yageo.com/documents/recent/PYu-RC_Group_51_RoHS_L_7.pdf	1	Resistors
1X06VERT-MALE	http://www.sullinscorp.com/catalogs/77_PAGE108-109_.100_MALE_HDR.pdf	2	
1X08VERT-MALE	http://www.sullinscorp.com/catalogs/77_PAGE108-109_.100_MALE_HDR.pdf	2	
POTENTIOMETERNO-MARKINGS_VERT-FEMALE	http://www.bourns.com/docs/Product-Datasheets/PDB18.pdf	4	
CAP_POLARIZED_ALUM-ELECTROLYTIC_SMD_220UF-16V-20%	https://industrial.panasonic.com/cdbs/www-data/pdf/RDE0000/ABA0000C1240.pdf	1	Polarized Capacitors
RESISTORSMD-0805_1/8W_2K_5%	http://www.yageo.com/documents/recent/PYu-RC_Group_51_RoHS_L_7.pdf	1	Resistors
2PT_GND_TIE		1	Dual Ground Supply
SWITCH_MOMENTARY_SPSTSPST-PTH-L12-W12-H7.2	https://www.sparkfun.com/datasheets/Components/Buttons/TSA12110%20TACT%20SWITCH.jpg	2	
SWITCH_SPDT5A-28V_13L-7W	https://www.e-switch.com/system/asset/product_line/data_sheet/129/100.pdf	1	
BATTERYPTH_HOLES		1	Generic Dual SMD Pad for Bettery Soldering
LED_SMD-805_DISCRETE_BLUE	http://optoelectronics.liteon.com/upload/download/DS-22-99-0226/LTST-C170TBKT.pdf	1	
LED_SMD-805_DISCRETE_GREEN	http://optoelectronics.liteon.com/upload/download/DS22-2000-073/LTST-C170KGKT.pdf	1	
SERVOMG90S_NO-MARKINGS	http://www.towerpro.com.tw/product/mg90s-3/	4	
LED_SMD-805_DISCRETE_RED	http://optoelectronics.liteon.com/upload/download/DS-22-99-0150/LTST-C170KRKT.pdf	1	

CHALLENGE #1: PCB DESIGN WITH EAGLE

Overview: From the previous robotic arm circuit, you are going to add 2 extra buttons, 2 LEDs, 4 resistors, 3 smoothing capacitors, 4 decoupling capacitors, 2 voltage regulators, 2 diodes, a LiPo connector, and a jack connector. Again, one button is going to be the ‘record’ button, where pressing it will put the robotic arm in recording mode. The other is the ‘play’ button, where pressing it will initiate the arm to replay the set of motions that it has previously recorded in recording mode. Moreover, the red LED is a visual indicator of the recording state, where it will be on when the arm is recording, and the green LED is a visual indicator of the playing state. The two resistors of 10K ohm will be used as pull-down resistor for the buttons, and the other two resistors are part of the LED setup. Lastly, the rest of components will allow the robotic arm to be power with 3 different power sources: a 6V 4xAA battery case (from part 2), a 7.4V LiPo battery, and 9V power adapter.

Question (NEED CHECK-OFF FROM TA FOR POINT):
(Hint: answer these questions as you are creating your schematic file on EAGLE)
What is pull-down resistor? and why do you need it?

What is the point of smoothing capacitor and decoupling capacitor?

Why do we use a diode in conjunction with a voltage regulator?

Objective: You are going to design a custom PCB board for your robotic arm using EAGLE. The board will fix on top of an Arduino Uno like a shield. You will need to submit both schematic, and board files checkoff credit.

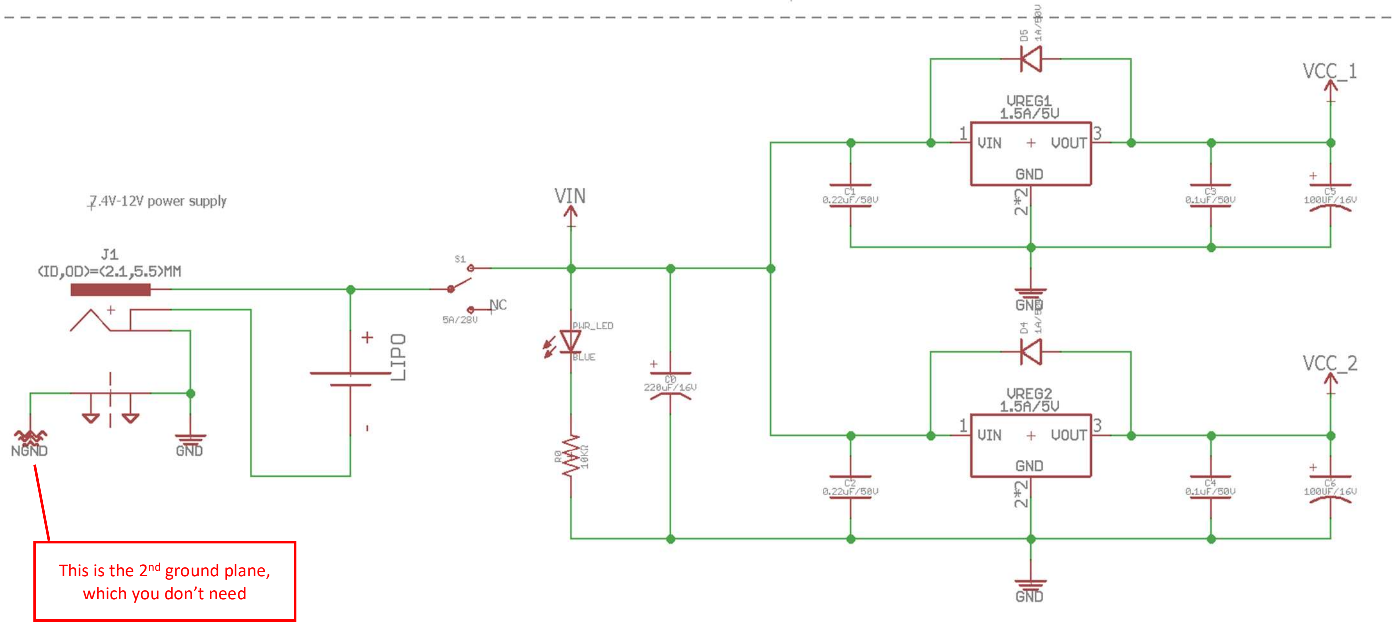
Note: a schematic of the board will be provided to facilitate your designing process, and all the necessary components are already in your PIB EAGLE library. Lastly, you don’t have to create two ground planes (NGDN, GND), and please speak with a TA for clarification if needed.

General steps:

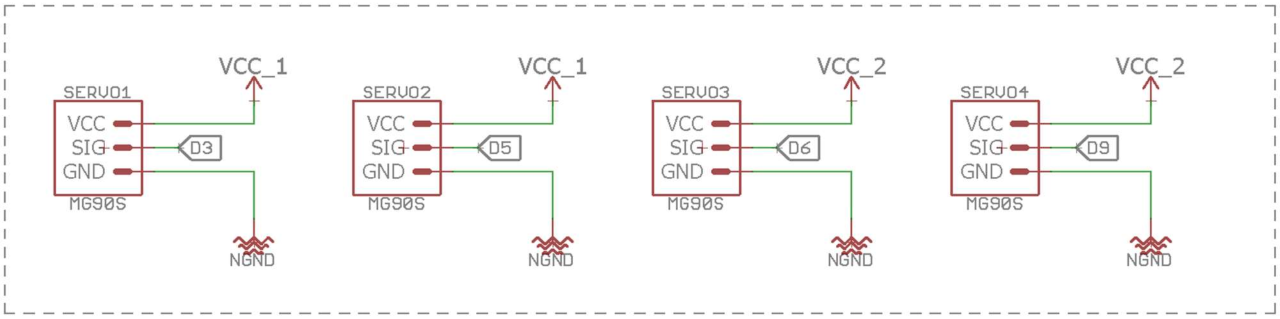
1. Following the schematic below, create the same schematic on EAGLE, and the ‘required project part’ list will be very helpful with locating all the components for your schematic. Again, you only need to create one ground plane (GDN). The other ground plane (NGND) is created for extra noise isolation between the servos and potentiometers.
2. Design your PCB board. A board design is provided, but you are free to change the components layout. The board dimension is 53.5mm x 56mm (hint: look at the actual PCB board for layout specific, since it has detail label of each component on the board.)
3. Get checkoff, and Submit both schematic and board files

Note: you do not have to design an exact board as shown, and please ask TA for assistance if needed

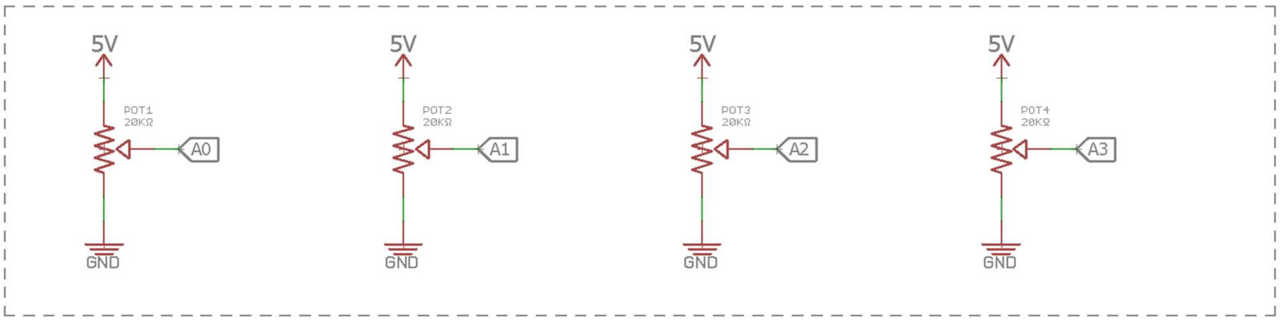
POWER DISTRIBUTION



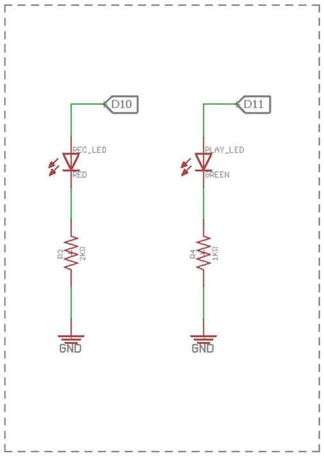
SERVOS



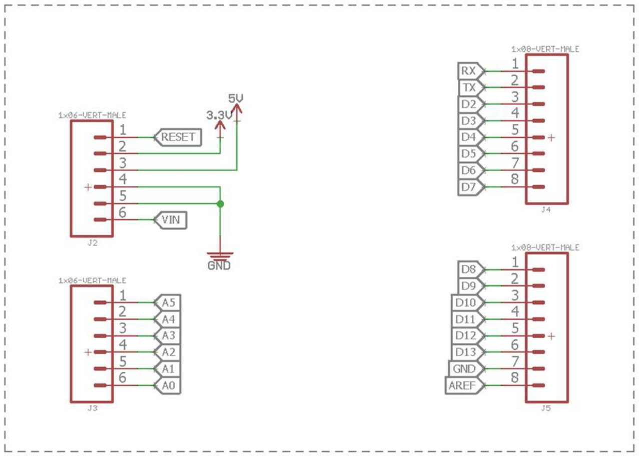
POTENTIOMETERS



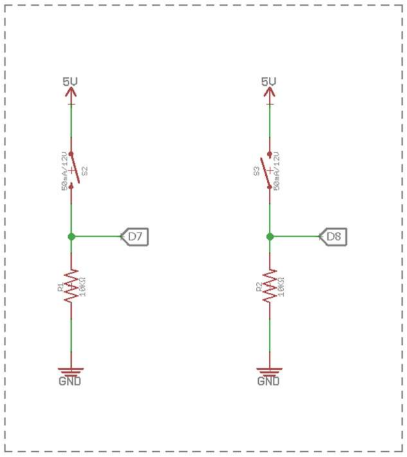
LEDS

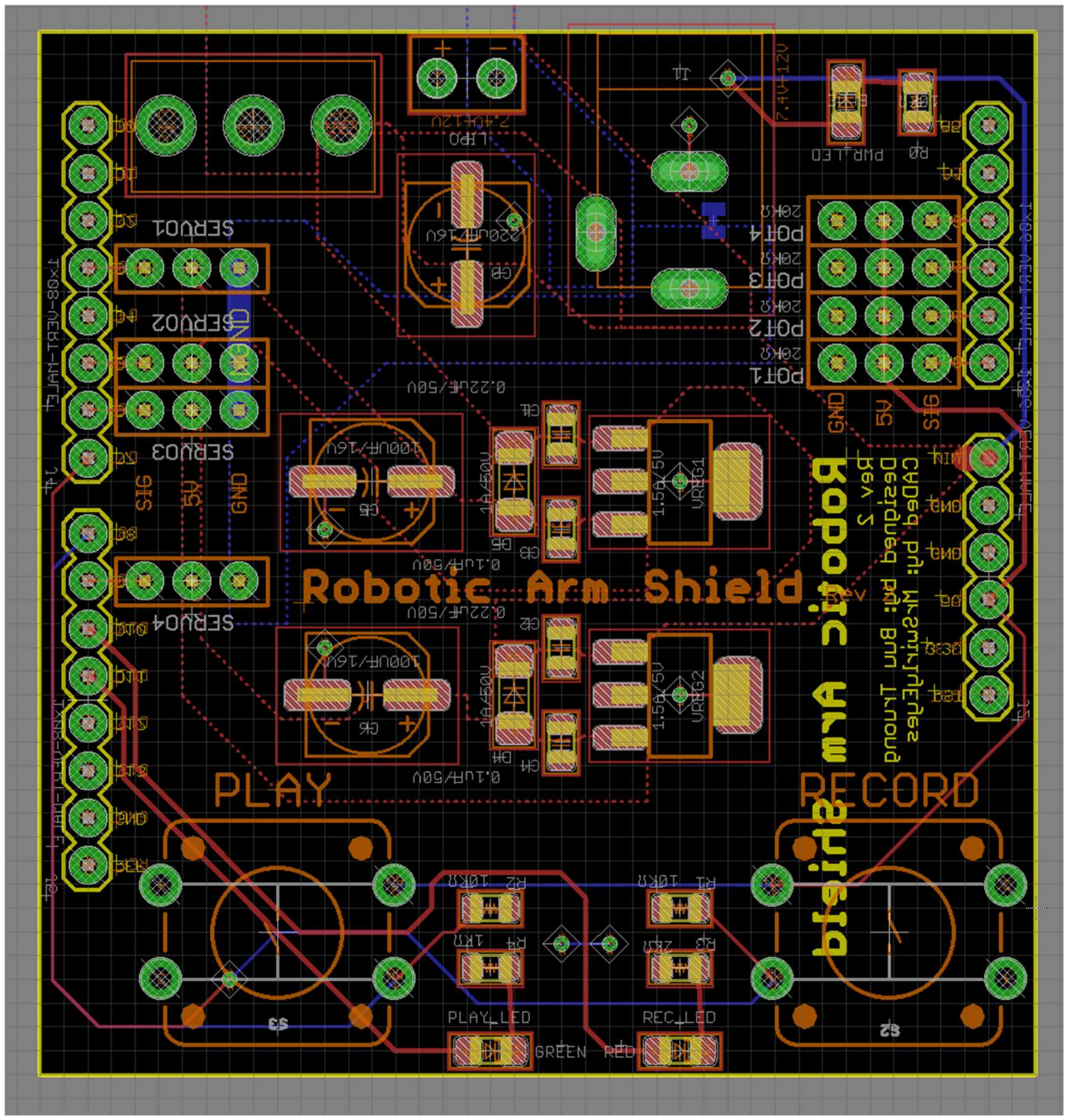


ARDUINO UNO



BUTTONS





CHALLENGE #2: SMD/THD SOLDERING OF PCB BOARD

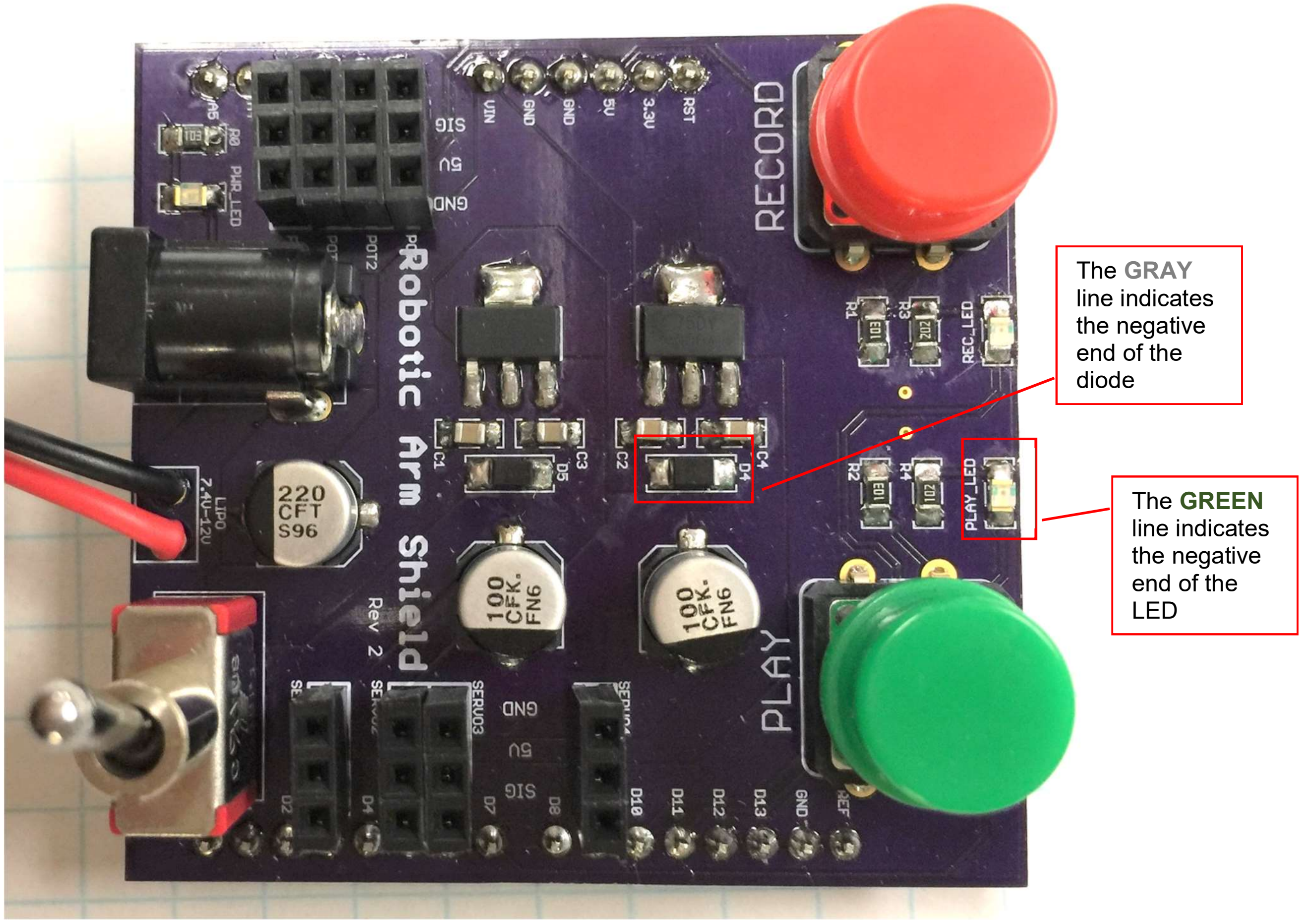
Objective: you are going to solder the PCD board that you have just designed. The board has both surface-mount and through-hole components. If you are unfamiliar with SMD soldering technique, please ask a TA to show you how to solder a surface-mount component.

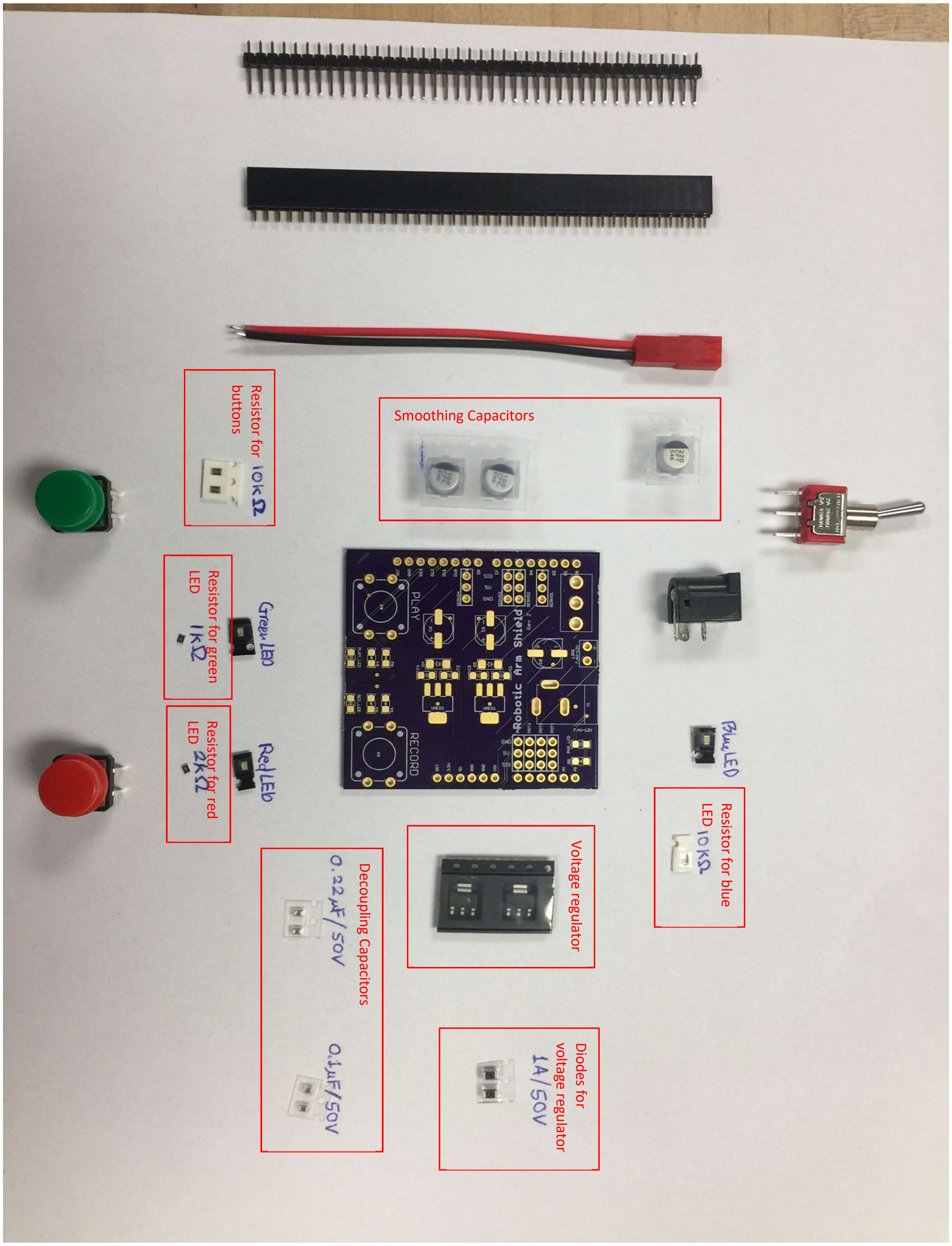
Note: Most of the components are not given to you initially because SMD parts are tiny and fragile. Thus, you should go get all the necessary SMD parts from a TA. A part-layout diagram is provided to help you keep track of all components. Also, a 7.4V LiPo battery or 9V power adaptor, and some female/male header pins will be given to you.

General steps:

1. Obtain all necessary components from TA following the diagram below. Some SMD parts look exactly like others, so if you mix them up, there is no way to tell them apart virtually. Therefore, using the part-layout diagram, you should tape the SMD component onto the diagram at appropriately labeled area of that specific part.
2. Solder the PCB board. A picture of a finished board is provided to help guide you through the soldering process, and you should use solder paste when soldering SMD components.
3. Double check all your connections and perform continuity test as needed.
4. Plug in the power source and make sure the PWR_LED is on.

Note: Positive-end and negative-end placement orientations matter for diodes and LED.


The GRAY line indicates the negative end of the diode
The GREEN line indicates the negative end of the LED
Robotic Arm Shield
Rev 2
RECORD
PLAY
PWR_LED
REC_LED
PLAY_LED
LIPO 7.4V-12V
VIN
GND
GND
5V
3.3V
RST
SIG
5V
GND


Resistor for buttons
10kΩ
Smoothing Capacitors
Resistor for green LED
1kΩ
Green LED
Resistor for red LED
2kΩ
Red LED
PLAY
RECORD
Robotic Arm Shield
Blue LED
Resistor for blue LED
10kΩ
Voltage regulator
Decoupling Capacitors
0.22µF/50V
0.1µF/50V
Diodes for voltage regulator
1A/50V

CHALLENGE #3: PROGRAM AUTOMATION MODE

Introduction: Protothread is a programing technique that allows a processor with a single core, like what you would find on Arduino board, to simulate multithreading algorithm. Before we continue our discussion on protothread, we should understand how multithreading functions. Multithreading is a programing technique that allows a processor with multi-core to process multiple programs at the same time, where each core in the processor is responsible for running one program. For example, a quad-core processor can run four different programs at the same time, parallel processing, because there are physically four different processor units on the chip. This is how we can listen to music on the computer while typing into the Microsoft word at the same time. Special exception, a single core processor with built-in program to divide up the physically single core into two or more virtual cores that can perform multithreading. This is called hyperthreading.

On the other hand, protothread do not divide the single core processor; rather, it utilizes a timer that allows it to switch (in microsecond interval) between multiple tasks and run them in a timely manner as though they are happening at the same time. For example, a protothread program that must read the temperature value from a sensor, store it on a SD card, and display the value on an LCD screen at the same time would read the temperature value at every 10 microseconds (ms), then the algorithm would switch to storing data at every 12 ms and, after that, then displaying the temperature value at every 14 ms. Most importantly, each task (thread) within the protothread program MUST run at constant time, **O(1)**, since any thread with longer time complexity would disrupt the timer. In Arduino, the millis function is used as a timer to enable protothreading. A library, TimedAction, is provided to help with implement of the millis function more cleanly and easily. **NOTE: you do not need to use this library to implement the millis function as a timer, but it is highly recommended.

Objective: Write an algorithm that implement protothread technique that enable your robotic arm to record position values and replay those values at a press of a button. Specifically, when record button is pressed, your algorithm will write positions obtained from potentiometers to servos and store them into arrays at the same time, and when play button is pressed, your algorithm will write those stored values to the servos continuously.

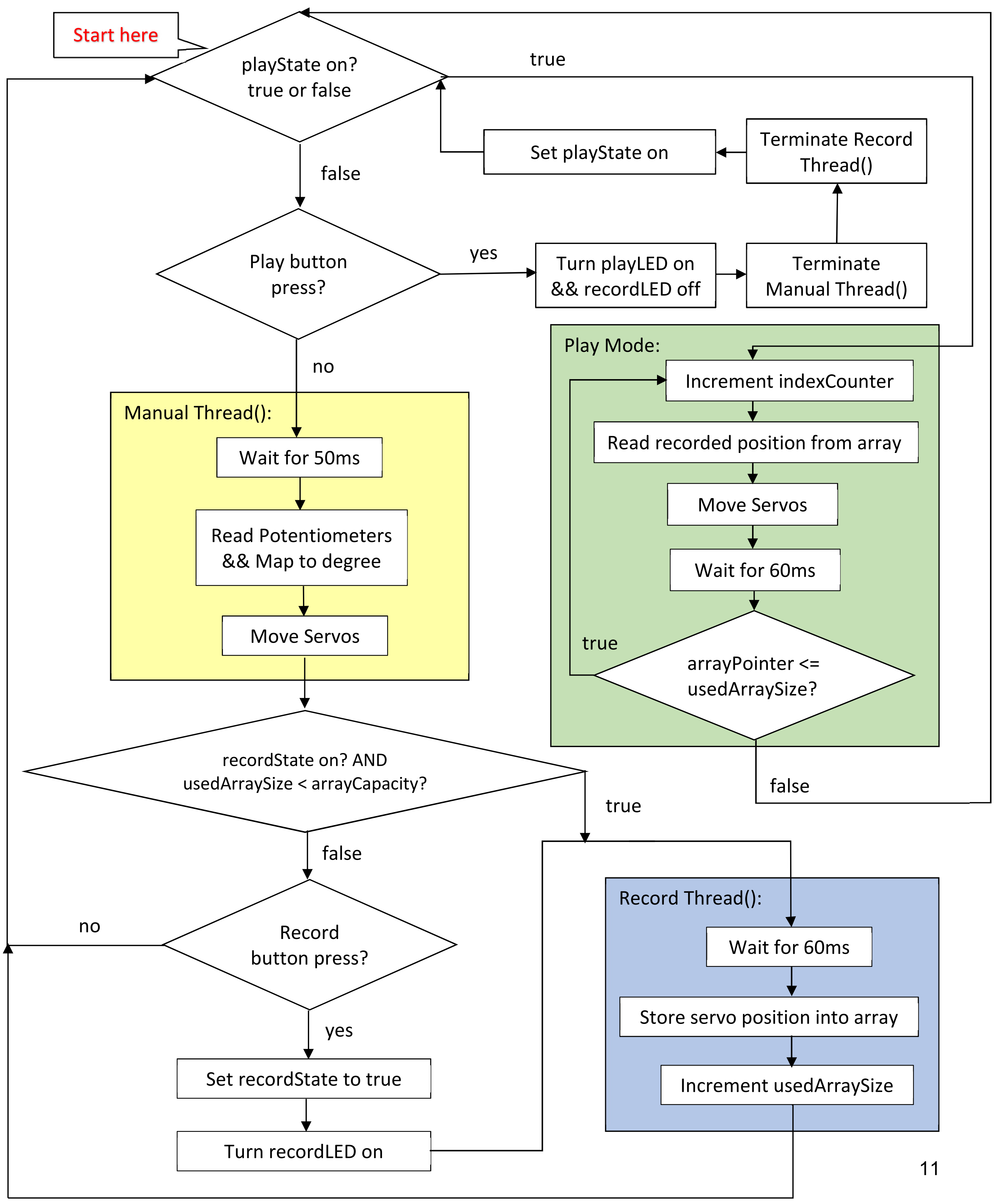
General Steps:

1. Import the TimedAction library into your local Arduino library, and the file can be found in the class Google Drive folder.
   Note: do not download the TimedAction directly off the documentation website, since it will not compile with current Arduino complier
2. Access the documentation website to look at the example code to learn how to use the TimedAction library: http://playground.arduino.cc/Code/TimedAction#Description
3. Follow the flowchart to program your automation code.
   Note: please look at the “definition of keywords in flowchart” to help with understanding of the flowchart

Hint:

- What is millis() function with respect to Arduino?
- What is the different between delay() and millis()?
- How many arrays are needed to store position values for four servos?
- How do you implement multiple millis() functions?

Follow flowchart to program the automation function: play and record state initialized as false.


Start here
playState on? true or false
true
false
Set playState on
Terminate Record Thread()
Play button press?
yes
Turn playLED on && recordLED off
Terminate Manual Thread()
no
Manual Thread():
Wait for 50ms
Read Potentiometers && Map to degree
Move Servos
Play Mode:
Increment indexCounter
Read recorded position from array
Move Servos
Wait for 60ms
true
arrayPointer <= usedArraySize?
false
recordState on? AND usedArraySize < arrayCapacity?
true
false
no
Record button press?
yes
Set recordState to true
Turn recordLED on
Record Thread():
Wait for 60ms
Store servo position into array
Increment usedArraySize

Definition of keywords in flowchart:

- playState is the state of the program, where the program is currently writing the stored position values in arrays to servos.
- record button is a button that is used to initiate the process of storing position values obtained from potentiometers into arrays.
- play button is a button that is used to initiate the process of writing the stored position values in arrays to the servos.
- recordState is the state of the program, where the program is currently storing the position values obtained from potentiometers and writing those values to the servos at the same time.
- usedArraySize is a counter that keeps track of the already allocated index in an array.
- arrayCapacity is a value that indicates the maximum of elements an array can hold.
- recordLED a LED that give visual indication of the recordState.
- playLED is a LED that give visual indication of the playState.
- indexCounter is a counter that indicates the current index of an array. This is used exclusively in play mode.

**DISCLAIMER: multicore processor does implement an advanced version of protothreading to execute multiple programs in the background.

CHALLENGE #4: MEMORY MANAGEMENT

Introduction: This automation algorithm will utilize all the Arduino Uno dynamic memory (RAM) because the position values are stored in RAM space. Therefore, the more RAM space you have, the more position values you can store, leading to a longer recording time. An easy way to expand RAM memory is efficient allocation of memory.

Objective: Look at your algorithm, and change the **type** (unit of information) that is used to define your variables (i.e. int, short, byte)

General Steps:

1. Find out the biggest whole number that position value (variable) can be
2. Determine the best unit of information (type) to represent that value. For example, if the biggest whole number of your position value is 100, then you should not use an **int** type to define your position variable because an **int** is 16 bits, which can represent value up to 2,147,483,647. So, you will waste a lot of memory when using an int type to represent 100.
3. Change the unit of information to the optimal type.
4. Repeat step 1 to 3 for other variables in your algorithm.

Please email me at buutruong1@gmail.com for any question or concern regarding this project