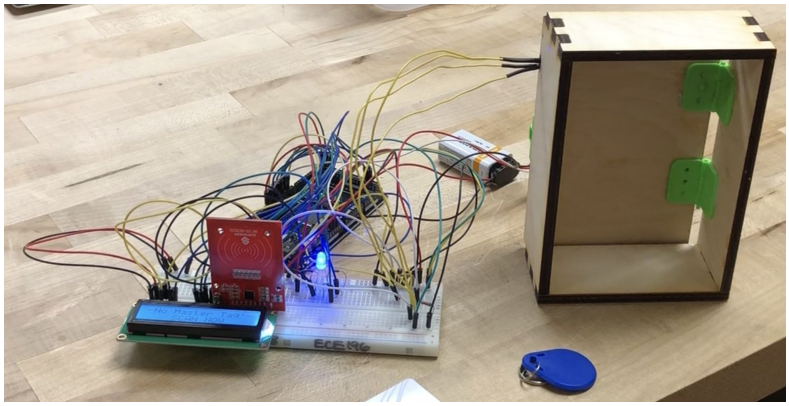
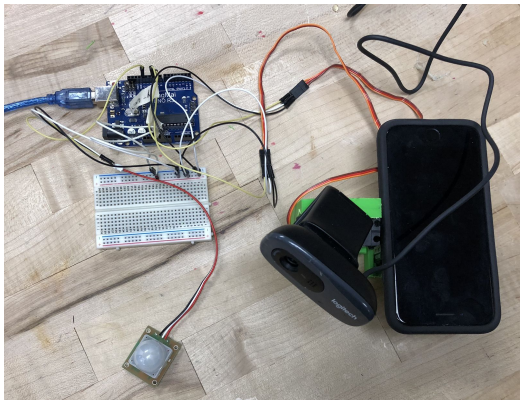
Version 2.0

June 15, 2018





Dual Security System

Team Security

Presented by: Joseph Chang
Hieu Nguyen
Richard Nguyen
Benjamin Chang
Kin Ming Loh

REQUIRED DOWNLOADS / INSTALLATION

1. Latest version of Arduino: https://www.arduino.cc/en/Main/Software
2. Latest version of MATLAB: https://matlab.ucsd.edu/student.html#access
3. Latest version of Solidworks: https://acms.ucsd.edu/services/software/available-software/solidworks.html
4. Latest version of Inkscape: https://inkscape.org/en/release/0.92.3/
5. RFID Arduino Library: https://github.com/miguelbalboa/rfid
6. Image Acquisition Toolbox: https://www.mathworks.com/help/matlab/matlab_env/get-add-ons.html
7. Computer Vision Toolbox: https://www.mathworks.com/help/matlab/matlab_env/get-add-ons.html

PROJECT DESCRIPTION & MOTIVATION

This project builds an electronic security system for a door or safe. A model door will be laser cut from wood and a latch lock will be attached to it. Two door hinges will be 3D printed. A master card will be able to add new cards by storing their IDs in a database. An LCD display shows commands such as “Scan card”, “Authorized”, or “Unauthorized”. If an unknown card is scanned after the master card, it will be added. If it is scanned after the master card again, it will be removed.

If an authorized card is scanned, a servo unlocks the latch lock. An RGB LED lights up green or red based on whether unlocking is successful or not. A proximity sensor senses the doors’ proximity to the door frame. If the door is opened then closed, the servo will relock the door.

When a PIR sensor detects motion, the webcam turns on and tracks the entrant’s face in real-time using two servos and MATLAB’s Computer Vision toolbox. The webcam will record video while motion is detected and store it in a computer. Parts are 3D-printed to connect the servos and webcam together.

This security system can be used to increase the security of a typical latch lock through RFID and surveillance. Using an RFID scanner avoids lockpicking and installing a camera has been proven to decrease crime by 40%. The camera’s stored surveillance can be reviewed after a crime as well. This system is relatively inexpensive to implement and could be installed homes or public environments.

PROJECT ORIGINALITY

This project builds upon an existing RFID scanner project by adding an RGB LED as a visual cue to tell whether an ID scan is successful or not. It also incorporates a camera controlled by a robotic arm. This system turns on when motion is detected and tracks a person’s face while recording video. When motion stops, this video is then stored in a computer. Using a motion detector lowers power consumption and memory overuse as the camera only records events of significance.

OVERALL LEARNING OBJECTIVES

- Matlab-Arduino serial communications
- Matlab-Webcam interface
- Servo calibration for face tracking
- Program RFID with Arduino IDE
- 3D Printing, Solidworks
- Laser Cutting, Inkscape

ALL REQUIRED PROJECT PARTS

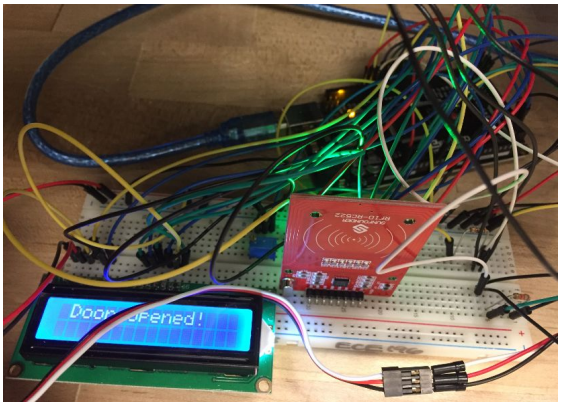
Material	Vendor	Price	Quantity	Cost	Grand Total
Dingmai Uno R3	Amazon	$10.99	1	$10.99	
Arduino Mega	Self	$0.00	1	$0.00	
Qunqi MFRC-522 RC522	Amazon	$7.29	1	$7.29	
CNY70 Optical Sensor	Amazon	$4.22	1	$4.22	
Small Servo Motor	Sparkfun	$8.95	1	$8.95	
RGB LED Motor	Sparkfun	$1.95	1	$1.95	
Arducam 1602 16x2 LCD Display	Amazon	$5.99	1	$5.99	
Logitech C270 Webcam	Amazon	$21.99	1	$21.99	
PIR Motion Sensor (JST)	Sparkfun	$9.95	1	$9.95	
Servo - Generic High Torque Standard Size	Sparkfun	$12.95	2	$25.90	
AK672/2-1 Cable	Digi-Key	$2.05	2	$4.10	
Breadboard Solderless 400 Tie	Digi-Key	$4.50	1	$4.50	
220 Ohm Resistor	Lab	$0.00	4	$0.00	
10k Ohm Resistor	Lab	$0.00	1	$0.00	
¼" Plywood	Lab	$0.00	1	$0.00	
					$105.83

REQUIRED PROJECT TOOLS/EQUIPMENT

- Laptop Computer
- Phone (or other weight) to hold down webcam servos

PROJECT BUILD STEPS

CARD SCANNER



Build the circuit found under “Project Schematics” using an Arduino Mega, proximity sensor, small servo, RGB LED, LCD display, and a MFRC522 RFID reader.

Open the Arduino IDE and download the RFID Arduino Library found in “Required Downloads” section. Go to the Arduino toolbar and go to: Sketch > Include Library > Add .ZIP Library. Select the Arduino Library you just downloaded and add it. Your Arduino IDE is now able to use functions and methods that are in the MFRC522 library.

Obtain the RFID Arduino code from How To Mechatronics found in the “References” section and upload it to Arduino. Test to see that the circuit works except for the LED by following the steps on the LCD display. Make sure to keep the proximity sensor covered with a finger until an authorized card has been scanned. Then uncover and cover it again to simulate a door opening and closing after an authorized card has been scanned.

Before proceeding, go through the code to understand how it works as the next step will add changes to the code. The code changes can be implemented on your own using the information found under the **Code Changes** section. Another option is to use the code found under the **Final Code** section which can be used directly or as a reference.

Code Changes

Note that the servo may not rotate at the desired angle. For this project, the unlocking servo value was 100 and the initial locking servo value was 10. These values can be changed as needed.

```
myServo.write(100); // Unlocks the door
myServo.write(10); // Initial lock position of the servo motor
```

To make it easier to set the RGB LED’s color, create a **setColor** function at the bottom of your code.

Red is for unauthorized cards, Green is for authorized cards, and Blue is for when the system is waiting for a card to be scanned. Use the following code to setup the RGB LED and call the **setColor** function to change the RGB LED color.

Initialization Code:

```
const int redPin = 22;
const int greenPin = 24;
const int bluePin = 26;
int red = 0;
int green = 0;
int blue = 0;
```

Setup Code:

```
// Set the three LED Pins as outputs
pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);
```

Lastly, code must be added to prevent someone from removing the mastercard using the mastercard itself. Place the following code in the code logic after the mastercard has been scanned and another card has just been scanned, but has not been added or removed yet.

```
if(tagID==myTags[0]){
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print(" Cannot remove  ");
   lcd.setCursor(0,1);
   lcd.print("  Master Tag!   ");
   printNormalModeMessage();
   correctTag=true;
   successRead=false;
   return;
}
```

Final Code

```
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#define RST_PIN   9
#define SS_PIN    10
byte readCard[4];
char* myTags[100] = {};
int tagsCount = 0;
String tagID = "";
boolean successRead = false;
boolean correctTag = false;
int proximitySensor;
boolean doorOpened = false;
```

```
const int redPin = 22;
const int greenPin = 24;
const int bluePin = 26;
int red = 0;
int green = 0;
int blue = 0;
// Create instances
MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal lcd(2, 3, 4, 5, 6, 7); //Parameters: (rs, enable, d4, d5, d6, d7)
Servo myServo; // Servo motor
void setup() {
  // Set the three LED Pins as outputs
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  setColor(0,0,255);
  // Initiating
  SPI.begin();        // SPI bus
  mfrc522.PCD_Init(); //  MFRC522
  lcd.begin(16, 2);   // LCD screen
  myServo.attach(8);  // Servo motor
  myServo.write(10); // Initial lock position of the servo motor
  // Prints the initial message
  lcd.print("-No Master Tag!-");
  lcd.setCursor(0, 1);
  lcd.print("    SCAN NOW");
  // Waits until a master card is scanned
  while (!successRead) {
    successRead = getID();
    if ( successRead == true) {
      myTags[tagsCount] = strdup(tagID.c_str()); // Sets the master tag into position 0
in the array
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Master Tag Set!");
      tagsCount++;
    }
  }
  successRead = false;
  printNormalModeMessage();
}
void loop() {
  int proximitySensor = analogRead(A0);
  // If door is closed...
  if (proximitySensor > 200) {
    if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader
continue
```

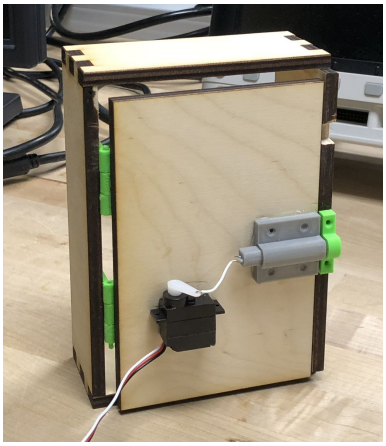
```
      return;
    }
    if ( ! mfrc522.PICC_ReadCardSerial()) {   //Since a PICC placed get Serial and
continue
      return;
    }
    tagID = "";
    // The MIFARE PICCs that we use have 4 byte UID
    for ( uint8_t i = 0; i < 4; i++) {  //
      readCard[i] = mfrc522.uid.uidByte[i];
      tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single
String variable
    }
    tagID.toUpperCase();
    mfrc522.PICC_HaltA(); // Stop reading
    correctTag = false;
    // Checks whether the scanned tag is the master tag
    if (tagID == myTags[0]) {
      lcd.clear();
      lcd.print("Program mode:");
      lcd.setCursor(0, 1);
      lcd.print("Add/Remove Tag");
      while (!successRead) {
        successRead = getID();
        if ( successRead == true) {
          if (tagID == myTags[0]){
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print(" Cannot remove  ");
            lcd.setCursor(0, 1);
            lcd.print("  Master Tag!   ");
            printNormalModeMessage();
            correctTag = true;
            successRead = false;
            return;
          }
          for (int i = 1; i < 100; i++) {
            if (tagID == myTags[i]) {
              myTags[i] = "";
              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("  Tag Removed!");
              printNormalModeMessage();
              return;
            }
          }
          myTags[tagsCount] = strdup(tagID.c_str());
```

```
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("   Tag Added!");
        printNormalModeMessage();
        tagsCount++;
        return;
      }
    }
  }
  successRead = false;
  // Checks whether the scanned tag is authorized
  for (int i = 1; i < 100; i++) {
    if (tagID == myTags[i]) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(" Access Granted!");
      setColor(0,255,0);
      myServo.write(100); // Unlocks the door
      printNormalModeMessage();
      correctTag = true;
    }
  }
  if (correctTag == false) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Access Denied!");
    setColor(255,0,0);
    printNormalModeMessage();
  }
}
// If door is open...
else {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Door Opened!");
  setColor(0,255,0);
  while (!doorOpened) {
    proximitySensor = analogRead(A0);
    if (proximitySensor > 200) {
      doorOpened = true;
    }
  }
  doorOpened = false;
  setColor(0,0,255);
  delay(500);
  myServo.write(10); // Locks the door
  printNormalModeMessage();
```

```
  }
}
uint8_t getID() {
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader
continue
    return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {   //Since a PICC placed get Serial and continue
    return 0;
  }
  tagID = "";
  for ( uint8_t i = 0; i < 4; i++) {  // The MIFARE PICCs that we use have 4 byte UID
    readCard[i] = mfrc522.uid.uidByte[i];
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single
String variable
  }
  tagID.toUpperCase();
  mfrc522.PICC_HaltA(); // Stop reading
  return 1;
}
void printNormalModeMessage() {
  delay(1500);
  lcd.clear();
  lcd.print("-Access Control-");
  lcd.setCursor(0, 1);
  lcd.print(" Scan Your Tag!");
}
void setColor(int red, int green, int blue){
 analogWrite(redPin, red);
 analogWrite(greenPin, green);
 analogWrite(bluePin, blue);
}
```

Once the code is changed, upload it to Arduino. The RFID system should have implemented the new changes. If not, revise the code or ask for help.

DOOR, FRAME, LATCH LOCK



Door & Door Frame
This model is designed in Inkscape and is made up of a four-sided door frame, a rectangular door, a latch latch, and two hinges. The height of the pieces are dependent on the wood thickness used.

Open Inkscape and create the door frame using the following dimensions for top, bottom, left, and right. Make four sketches of rectangle with the following dimensions:

a. Top: 112.7 mm L x 50 mm W
b. Bottom: 112.7 mm L x 50 mm W
c. Left: 50mm W x 162.7 mm L
d. Right: 58.44 mm W x 162.7mm L

Use *path* → *difference* to cut 6.35 mm W x 10 mm L blocks from two sides of the four pieces. For the right side of the door frame, cut out two 6.35 mm W x 18.44 mm L blocks and one 7.5 mm W x 7.5 mm L block. Finally, make a rectangular sketch of the door which is 96 mm W x 150 mm L. Laser cut your give pieces (top, bottom, left, right, and door) using ¼" plywood (¼ inch).

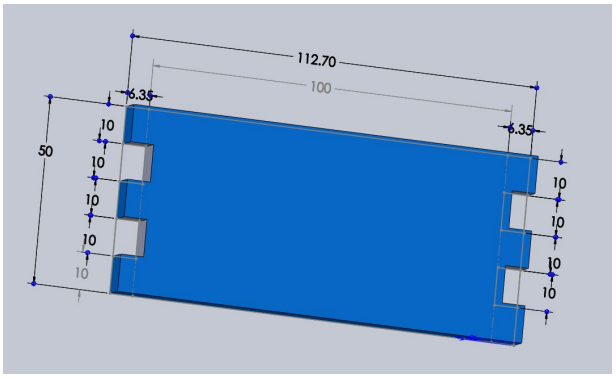


Figure 1 - Top and bottom door frame

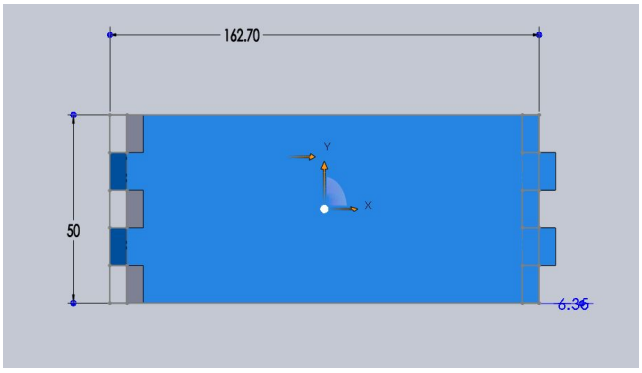


Figure 2 - Left door frame

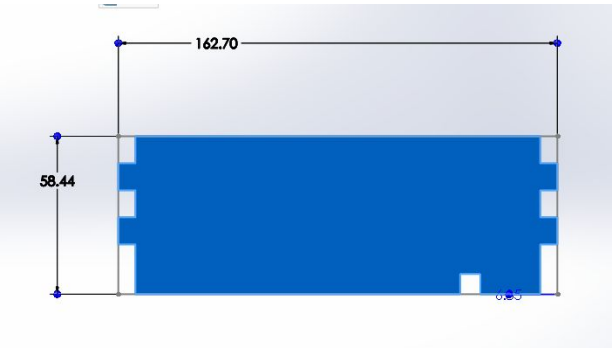


Figure 4 - Right door frame

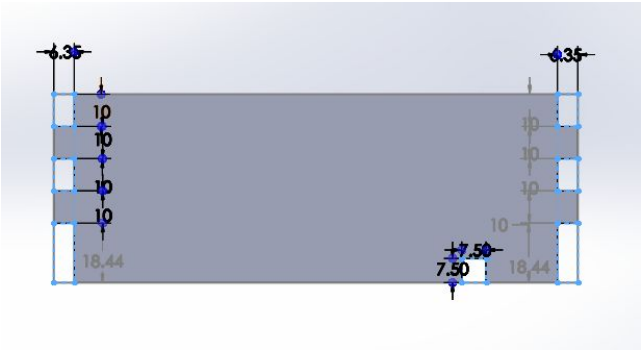


Figure 5 - Right door frame

Latch Base

In Solidworks, create the sketch shown in Figure 6. Use the boss extrude feature to give it a depth of 30mm.

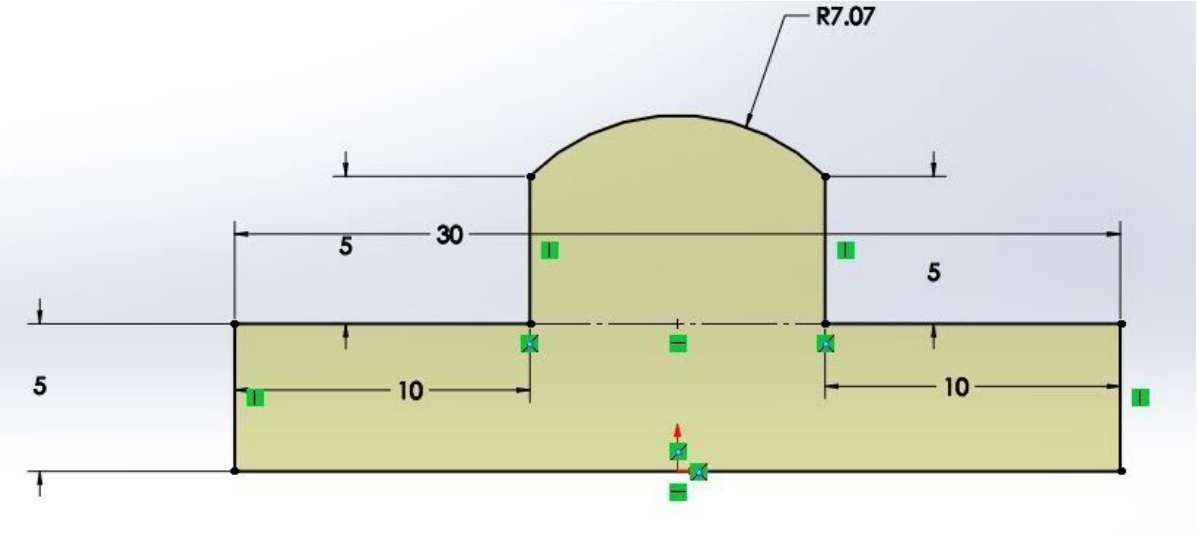


Figure 6 - Sketch of the latch base

Create a second sketch on the latch base as shown in Figure 7. Use the cut extrude feature to make a hole through the previous part with this shape.

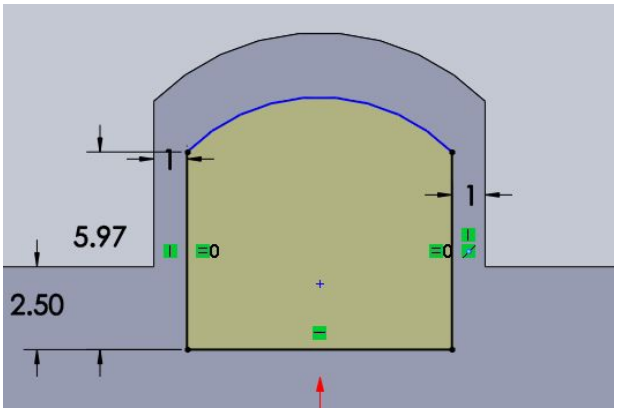


Figure 7 - Sketch to cut through the latch base

Finally, use the hole wizard to make four holes for screws. The hole size is M2.

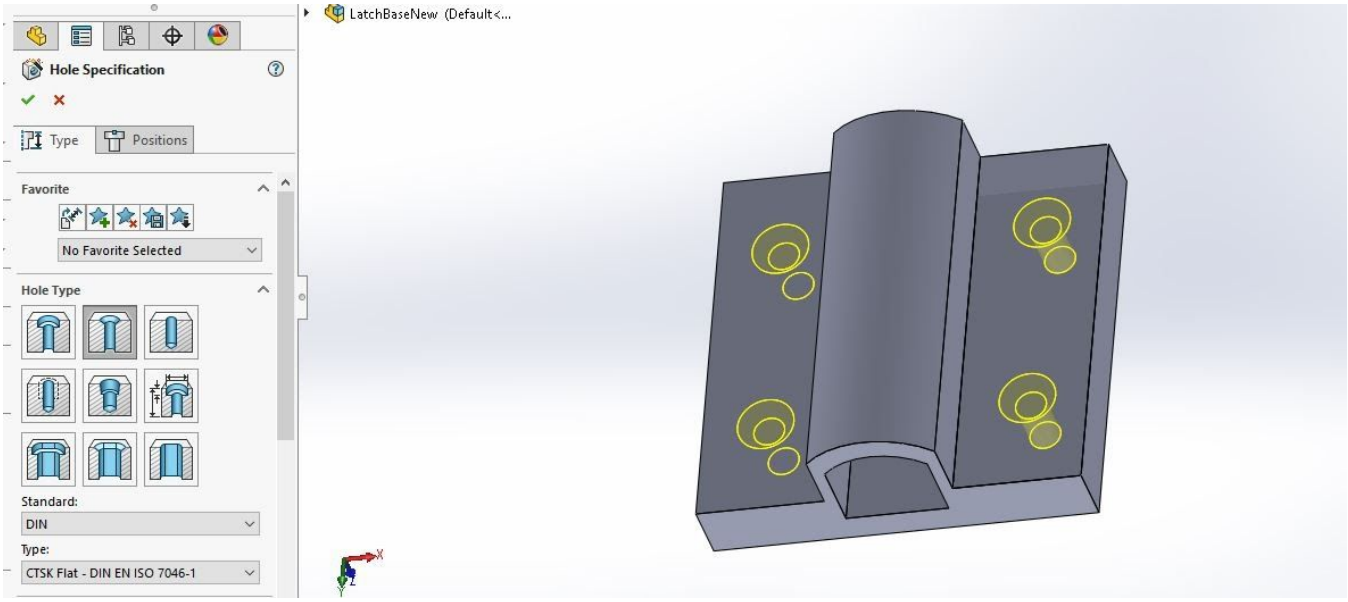


Figure 8 - Hole Wizard

Follow the similar steps above to make another part as shown in Figure 9.

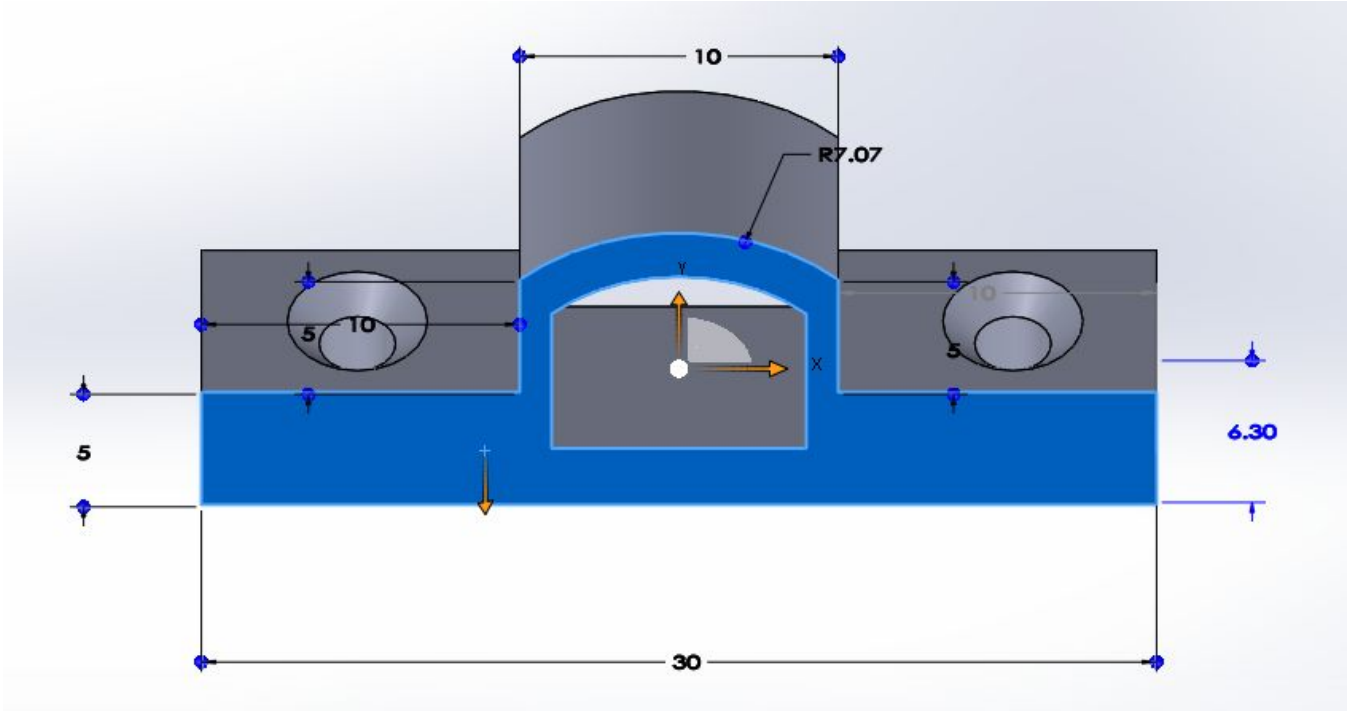


Figure 9 - Another part of latch attached to the side of the door

Latch Pin

In Solidworks, create a new sketch as shown in Figure 10. Use the boss extrude feature to it a depth which depends on how long the latch pin is. Now, make a loop at the end of the latch pin so the lock can be attached to a servo using a wire. To do this, make a circular sketch on the top plane and an arc sketch on the right or front plane. Combine the two sketches using the sweep feature.

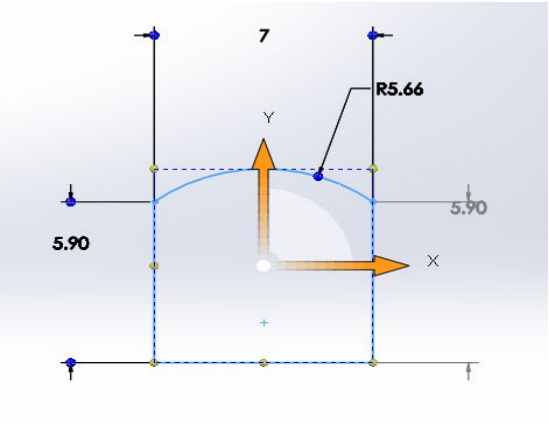


Figure 10 - Sketch of the latch pin

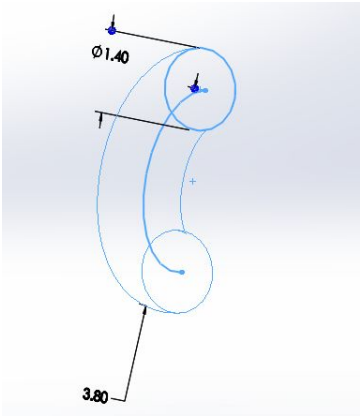


Figure 11 - Sketch of the loop

Hinge Design

There are three parts for each of the two identical hinges. Create a sketch as shown in Figure 12. Make copies of this sketch so there are four total. On two of these sketches, use the boss extrude feature to make the two-colored region in Figure 13 a 3D object. Then, make an extrude cut as shown in Figure 14. Use the hole wizard to create three holes on the large, flat side as shown in Figure 15. Finally, fillet both corners as shown in Figure 15.

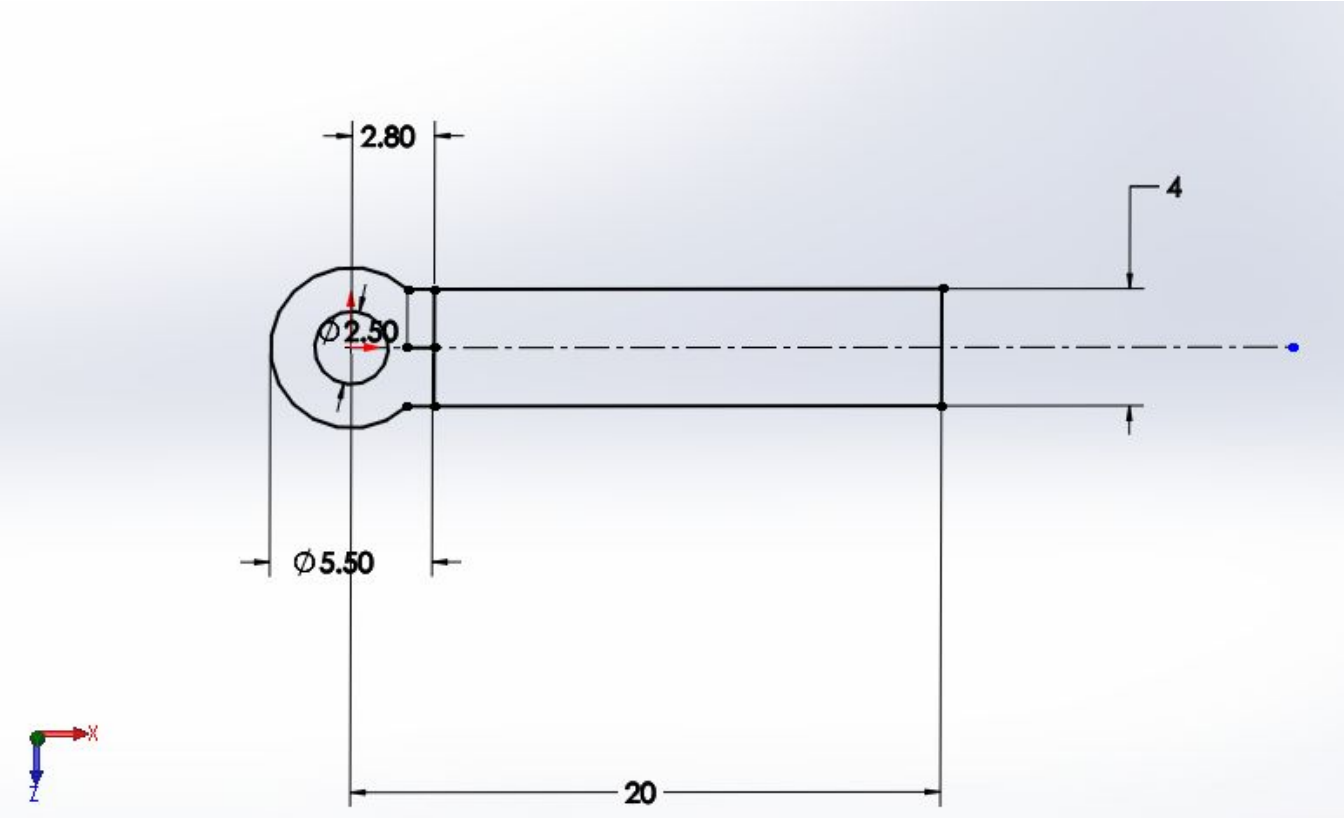


Figure 12 - Sketch for the hinge

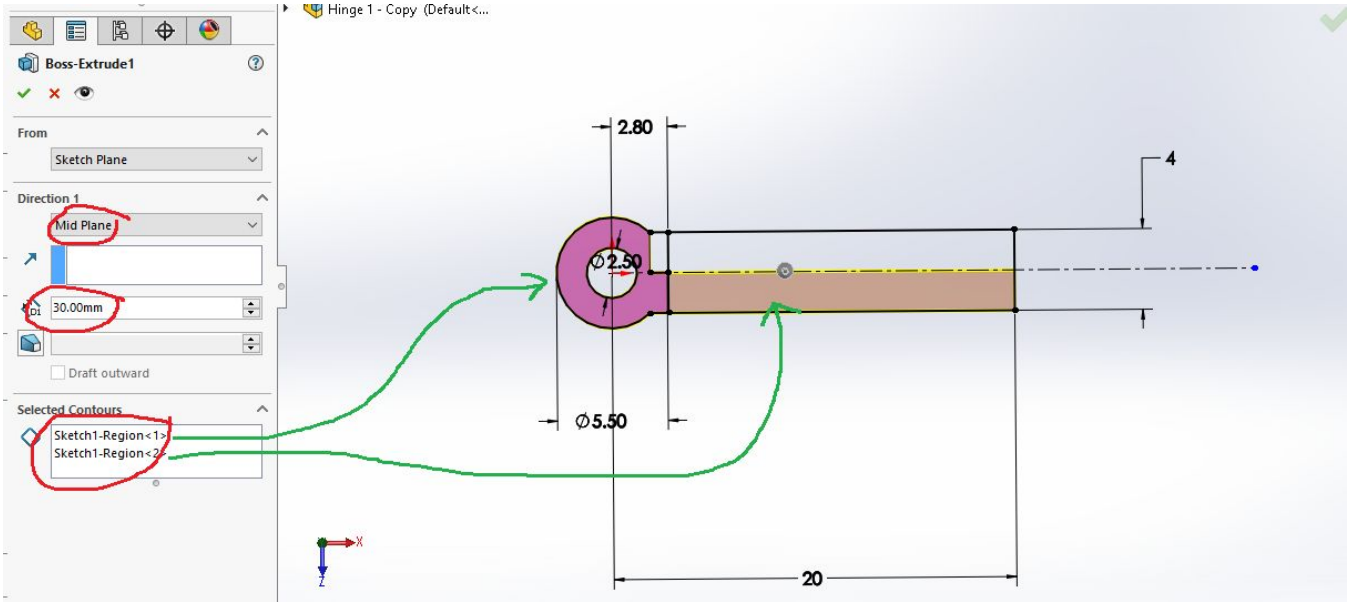


Figure 13 - Boss extrude

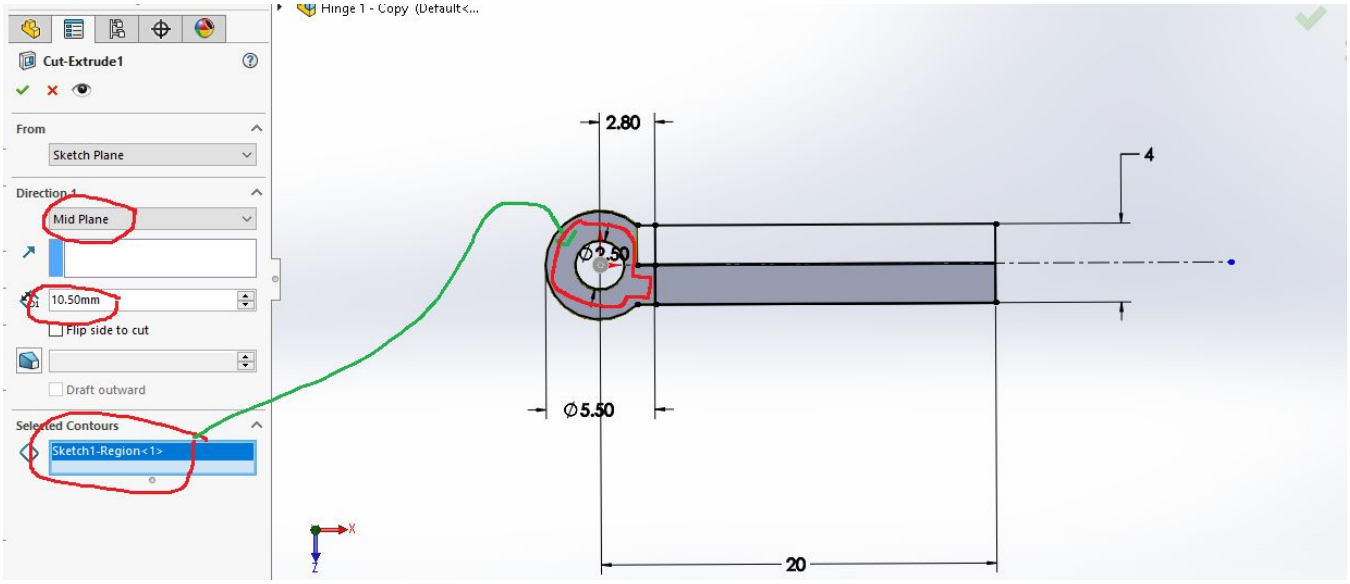


Figure 14 - Cut extrude

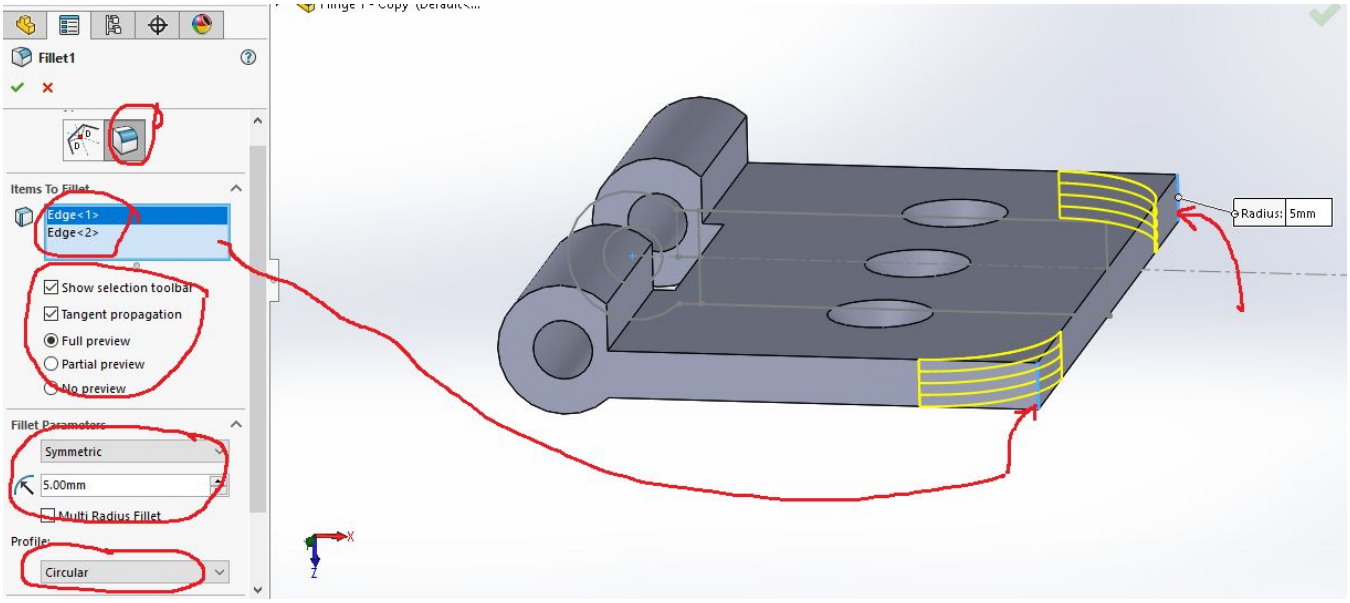


Figure 15 - Fillet

Hinge - Part 2

On the other two sketches, use the boss extrude feature to make the yellow region in Figure 16 a 3D object. User boss extrude again on the yellow region shown in Figure 17. Use the hole wizard to create three holes on the large flat side. Finally, use the fillet feature to curve the edge at the corner.

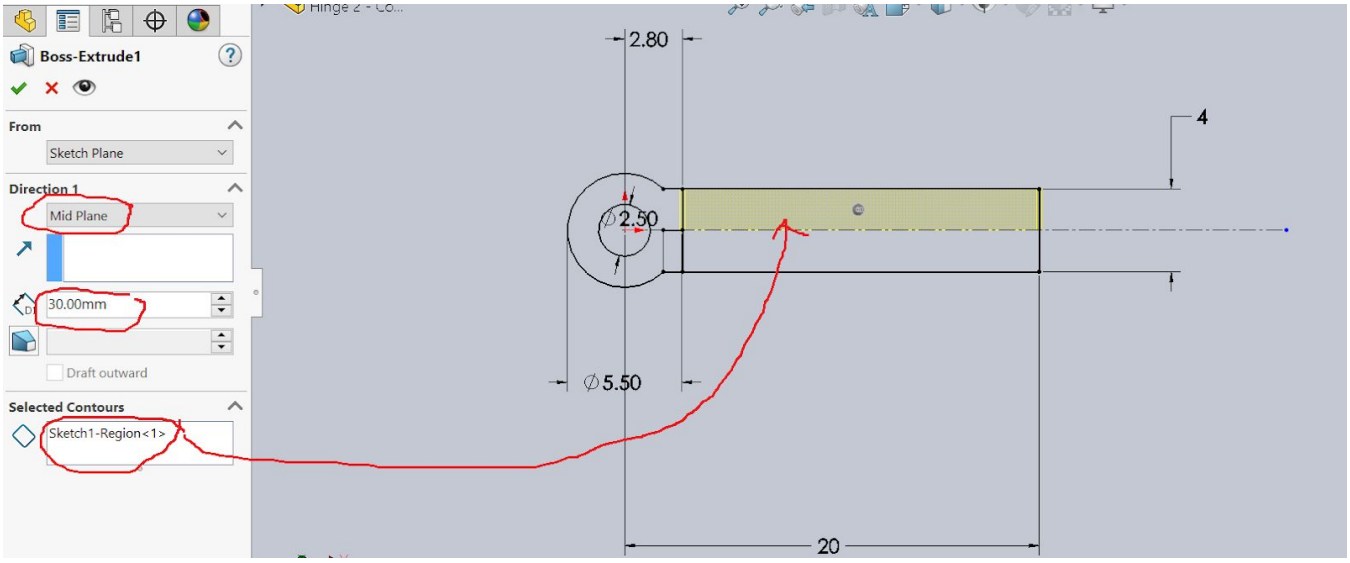


Figure 16 - Boss extrude

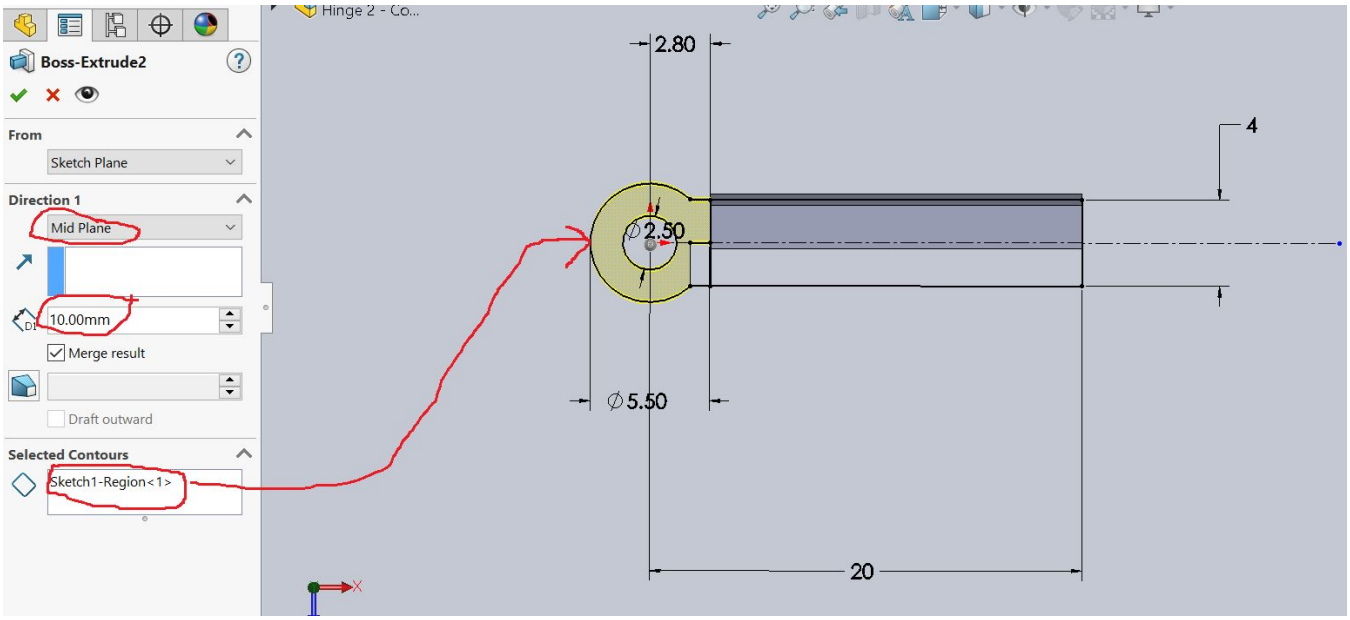


Figure 17 - Boss extrude

Hinge Pin

Create a circle sketch with radius 2 mm and make the sketch an 3D object with the same depth as the length of hinge. Use the fillet feature to curve the edge.

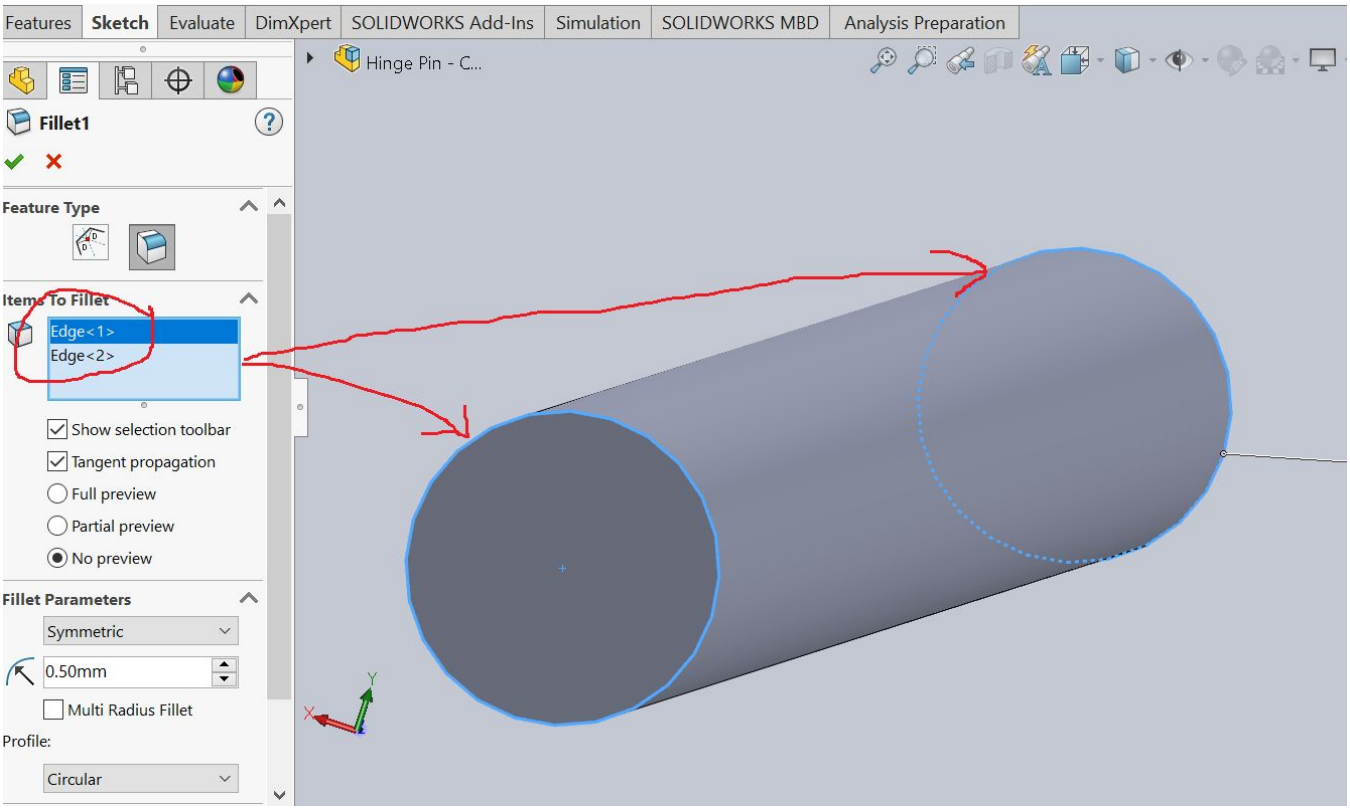


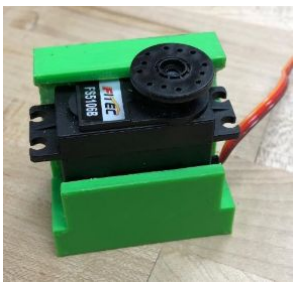
Figure 18 - Fillet of the pin

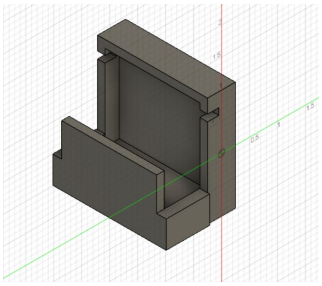
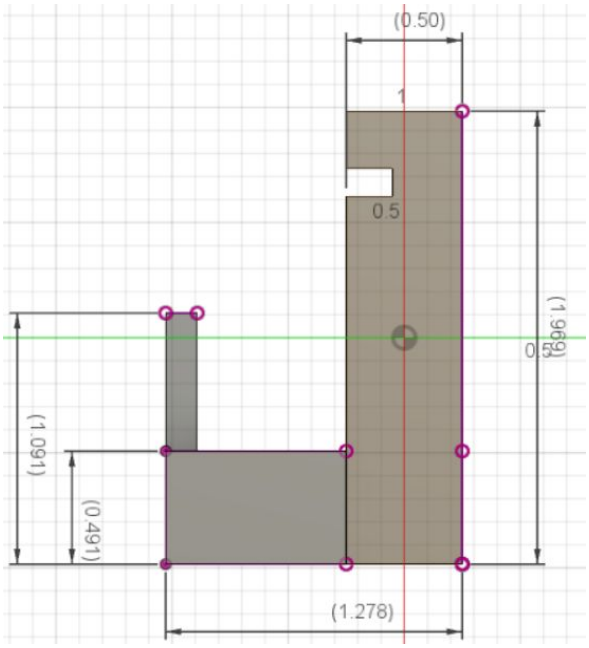
Now, 3D print the two latch base parts, latch pin, two hinges, and two hinge pins. Assemble the door, latch lock, and hinges using super glue.

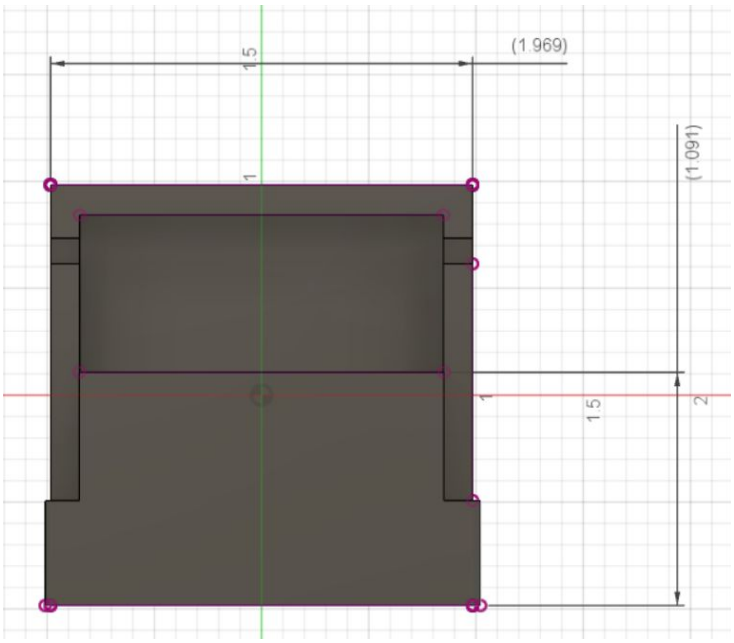
WEBCAM TRACKING

Build the circuit found under “Project Schematics” using an Arduino Uno, 2 standard servos, and a PIR motion sensor. Connect the x-axis servo to D6, the y-axis servo to D7, and the PIR motion sensor to D8.

Design a servo connector for the standard servos in Solidworks using the dimensions shown below. When finished, 3D print the parts and place the servos inside.





Download the Matlab code from Instructables found under the “References” section. Close any open serial ports at the beginning of the code using:

```
%% Close open serial ports
if ~isempty(instrfind)
fclose(instrfind);
delete(instrfind);
end
instrfind
```

Change the COM port to the port your Arduino is connected to.

```
a = serial('COM8','BaudRate',9600);
```

If your laptop has a built-in webcam, the device ID ‘1’ may need to be changed to ‘2’.

```
obj =imaq.VideoDevice('winvideo', 1, 'I420_320x240','ROI', [1 1 320 240]);
```

Now, implement a loop for the webcam to continuously read the motion sensor’s status from the Arduino serial port. It will turn on the webcam when motion is detected and send face center coordinates to Arduino. It will save frames as a file when motion is no longer detected. Note that saving frames is commented out as it slows performance. Replace all code after figure('menubar','none','tag','webcam'); with the following and take time to go through to the comments to understand how it works.

```
k = 1;
while k == 1; %infinite loop
    release(obj); %turn off webcam
    wait=0;
    ms = str2double(fgetl(a)); %read Arduino serial for motion sensor status
    i=0;
    while ms == 1; %motion detected
        i=i+1;
        while (wait<600)
            wait=wait+1
            ms = str2double(fgetl(a));
            %read Arduino serial for motion sensor status
            if ms == 0 %motion not detected
                wait = 700
                time = 60;
            end
            frame{i}=step(obj); %save frames in cells
            %STEP Acquires a single frame from image acquisition Device
            %frame is the Variable assigned to an image which is either RGB or
            %GRAYSCALE
            %Acquires a single frame from the VideoDevice System Object,obj.
            bbox=step(faceDetector,frame{i});
            if(~isempty(bbox)) %face detected
                centx=bbox(1) + (bbox(3)/2)
                centy=bbox(2) + (bbox(4)/2)
                c1=(centx);
                c2=(centy);
                fprintf(a,'%s',char(centx));
                %send face center x-coordinate to Arduino
                fprintf(a,'%s',char(centy));
                %send face center y-coordinate to Arduino
            end
            %BBOX=Bounding Box
            %step returns a Matrix of M-by-4 where M is some Variable to bbox
            %M defines bounding boxes containing the detected objects
```

```
%Each row in Matrix has 4 element Vector [x y width height] in pixels
%The objects are detected from Image Named as 'frame'
%detected objects are from face
boxInserter  = vision.ShapeInserter('BorderColor','Custom',...
      'CustomBorderColor',[255 0 255]);
%It inserts shapes according to matrix dimensions
%BorderColor is to specify the color of Shape by Default is Black
%Here We set it to 'Custom' so we can use 'CustomBorderColor' to specify
%the color of the border by vector representation
videoOut = step(boxInserter, frame{i},bbox);
%The Step function here returns an image
%Image consists of a Bounding box for the frame
%The BoxInserter inserts a frame around the image
%Output image is set to variable 'VideoOut'
imshow(videoOut,'border','tight');
%imshow basically displays images
%parameters 'Border','tight' indicates and compels the images to be
%displayed without a border
f=findobj('tag','webcam');
if (isempty(f))
[hueChannel,~,~] = rgb2hsv(frame{i});
% Display the Hue Channel data and draw the bounding box around the face.
%%figure, imshow(hueChannel), title('Hue channel data');
rectangle('Position',bbox(1,:),'LineWidth',2,'EdgeColor',[1 1 0])
%Creates 2-D rectangle at Position of BBOX with width and Edgecolor
hold off
%Resets to default behaviour
%Clears existing graphs and resets axis properties to their Defaults
noseDetector = vision.CascadeObjectDetector('Nose');
%Detects nose properties from the video frame using Cascade package
%the properties are assigned to a variable noseDetector
faceImage    = imcrop(frame{i},bbox);
%crops the Image 'Frame' with Bounding BOX
%%imshow(faceImage)
%Displays image
noseBBox     = step(noseDetector,faceImage);
%Returns NoseBBOX Matrix
noseBBox(1:1) = noseBBox(1:1) + bbox(1:1);
videoInfo    = info(obj);
ROI=get(obj,'ROI');
%returns the value of Specified property from the Obj image
VideoSize = [ROI(3) ROI(4)];
videoPlayer  = vision.VideoPlayer('Position',[300 300 VideoSize+60]);
%Play video or display image with specified position
tracker = vision.HistogramBasedTracker;
initializeObject(tracker, hueChannel, bbox);
time=0;
```

```
            while (time<60)
                time=time+1
                % Extract the next video frame
                frame{i} = step(obj);
                ms = str2double(fgetl(a)); % update ms
                if ms == 0 %motion not detected
                    wait = 700;
                    time = 60;
                    release(obj); %close webcam
                    release(videoPlayer);
                end
                % RGB -> HSV
                [hueChannel,~,~] = rgb2hsv(frame{i});
                % Track using the Hue channel data
                bbox = step(tracker, hueChannel);
                % Insert a bounding box around the object being tracked
                videoOut = step(boxInserter, frame{i}, bbox);
                %Insert text coordinates
                % Display the annotated video frame using the video player object
                step(videoPlayer, videoOut);
                pause (.2)
            end
            % Release resources
            %release(obj); %close webcam
            %release(videoPlayer);
            %release(vidobj);
            close(gcf)
            break
        end
        pause(0.05)
        end
%               if ms == 0 %motion not detected
%                   c = clock;
%                   save('c.mat','frame') %save frames as file
%               end
    end
end
```

Download the Arduino code from Instructables found under the “References” section. Add the following code for the motion sensor to detect motion and communicate whether motion was detected or not to Matlab.

```
const int MOTION_PIN = 8; // Pin connected to motion detector
pinMode(MOTION_PIN, INPUT_PULLUP);
int proximity = digitalRead(MOTION_PIN);
  if (proximity == LOW) //Motion detected
```

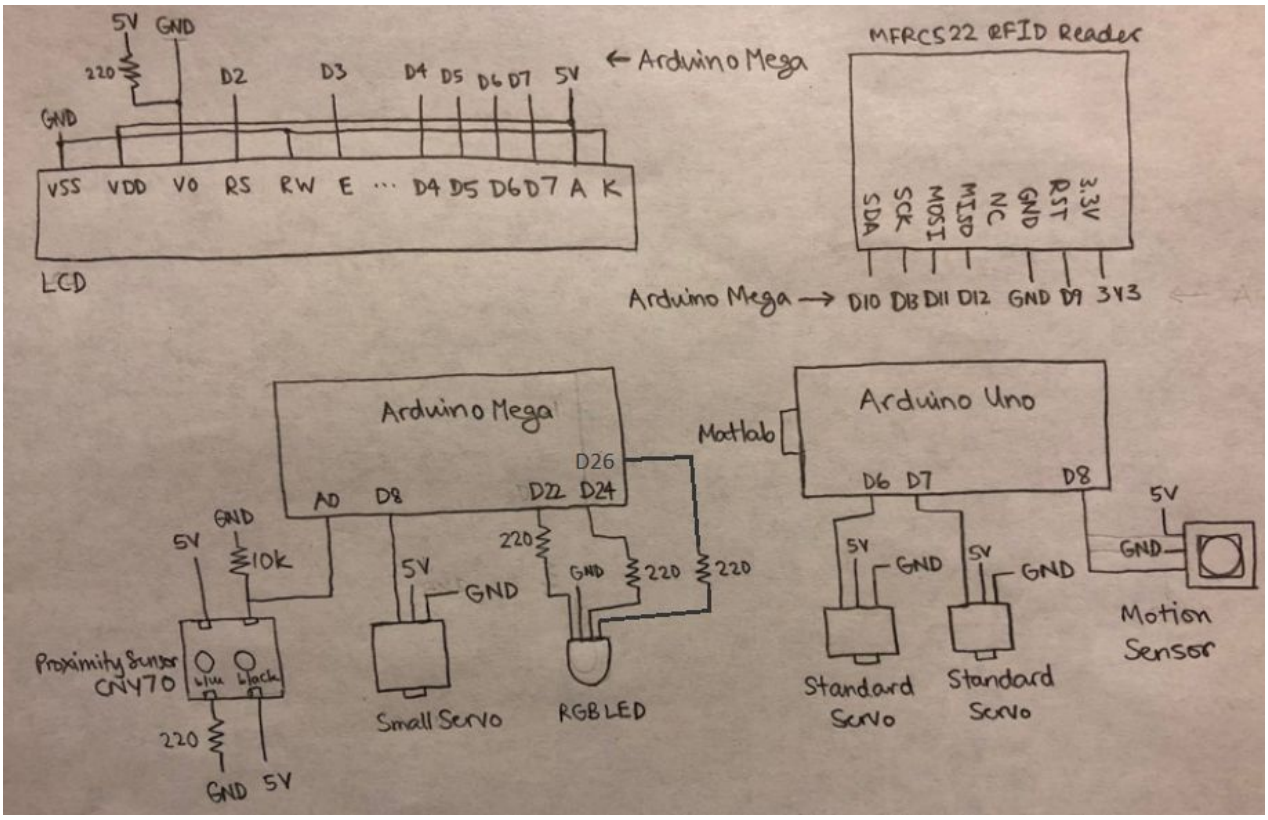
```
  {
   Serial.println("1");
  }
  else  //Motion not detected
  {
   Serial.println("0");
  }
  delay(100);
```

The x-axis servo tracked the face as expected, but the y-axis had a problem where it continues to turn downwards. To fix this issue, change the conditions with the following code:

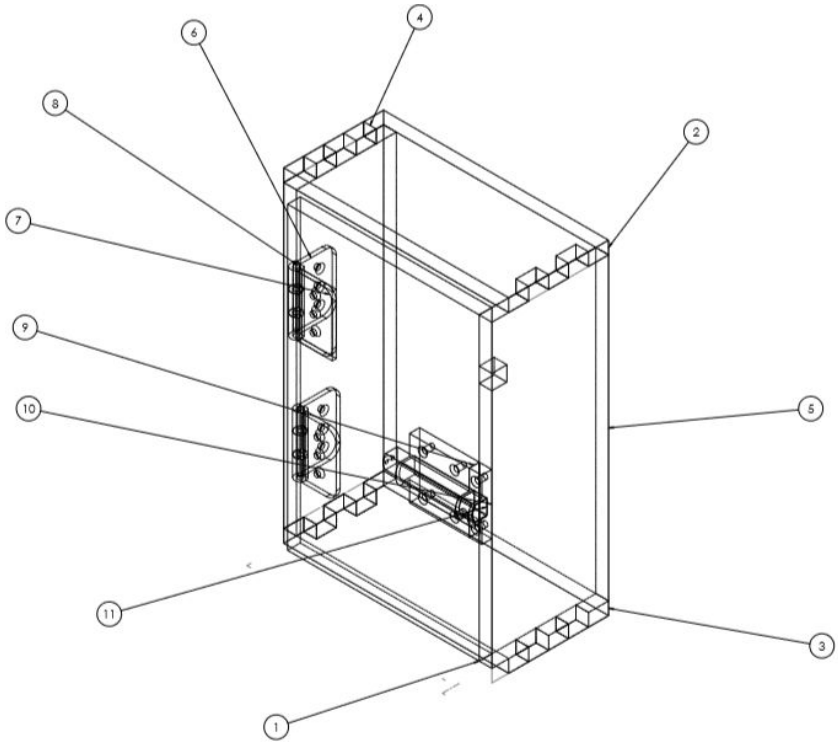
```
//Find out if the Y component of the face is below the middle of the screen.
if(valy < (100)){
  if(posy >= 5)posy += distancey; //If it is below the middle of the screen, update the
tilt position variable to lower the tilt servo.
}
//Find out if the Y component of the face is above the middle of the screen.
else if(valy > (100)){
  if(posy <= 175)posy -= distancey; //Update the tilt position variable to raise the tilt
servo.
}
```

Now the code is complete. Connect large gears to the top of your servos. Hot glue the servo connectors and webcam as shown below. Close any open serial ports by running the code at the beginning of this section in Matlab. Upload the Arduino code. Upload the Matlab code. Keep constant motion in front of the motion sensor for the webcam to stay on and tracking. Test your servos to see if track. If they do not, debug the code.

PROJECT SCHEMATICS

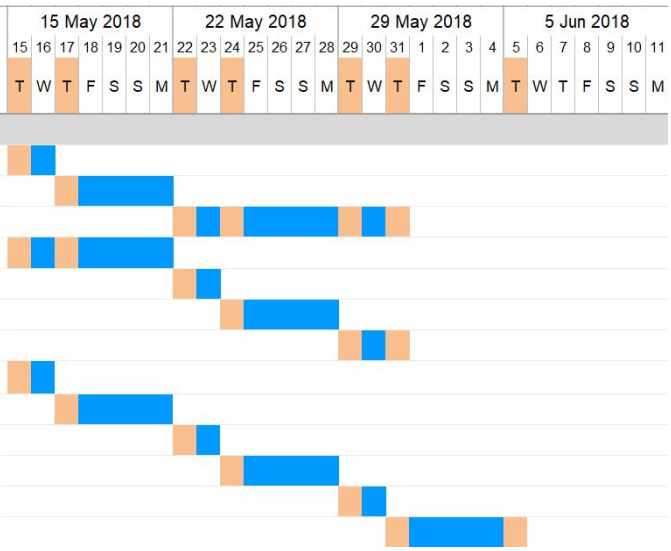


ITEM NO.	PART NUMBER	QTY.
1	door	1
2	Part1	1
3	Part2	1
4	Part3	1
5	Part4	1
6	Hinge 1	2
7	Hinge 2	2
8	Hinge Pin	2
9	LatchBaseNew	1
10	LatchPart2	1
11	LatchPin	1



PROJECT TIMELINE

WORK	TASK	START	END	DAYS
Card Scanner	Build Circuit	Tue 5/15/18	Wed 5/16/18	2
Card Scanner	Calibrate Proximity Sensor and Servo	Thu 5/17/18	Mon 5/21/18	5
Card Scanner	Program RFID, servo, led, lcd, sensor	Tue 5/22/18	Thu 5/31/18	10
Door	Laser Cut Door, Frame, Lock	Tue 5/15/18	Mon 5/21/18	7
Door	3D Print Hinges	Tue 5/22/18	Wed 5/23/18	2
Door	Assemble Door	Thu 5/24/18	Mon 5/28/18	5
Webcam	3D Print webcam servo connectors	Tue 5/29/18	Thu 5/31/18	3
Webcam	Build Circuit, Setup Arduino/Matlab	Tue 5/15/18	Wed 5/16/18	2
Webcam	Read Video Frames, Detect Face	Thu 5/17/18	Mon 5/21/18	5
Webcam	Track Face with Arduino Servos	Tue 5/22/18	Wed 5/23/18	2
Webcam	Record video and save as file	Thu 5/24/18	Mon 5/28/18	5
Webcam	Add Motion Detector	Tue 5/29/18	Thu 5/31/18	2
Debug	Make sure everything works	Fri 6/01/18	Wed 6/06/18	6



Name	Task
Joseph Chang	Webcam Tracking
Hieu Nguyen	Webcam Tracking
Richard Nguyen	Card Scanner
Benjamin Chang	Card Scanner
Kin Ming Loh	Door, Frame, Latch Lock

REFERENCES

1. How To Mechatronics is a good project-based resource for Arduino. Our project is based off their RFID door lock tutorial.
   https://howtomechatronics.com/tutorials/arduino/rfid-works-make-arduino-based-rfid-door-lock/

2. Instructables is a good project-based resource with a variety of platforms just like Adafruit. Our project is based off their Real Time Face Tracking tutorial.
   https://www.instructables.com/id/Real-Time-Face-Tracking-Robot-With-Arduino-and-Mat/

3. SparkFun is a vendor that sells a variety of platforms contains tutorials. They have a great tutorial on how to hookup a PIR motion sensor and write to it in Arduino.
   https://learn.sparkfun.com/tutorials/pir-motion-sensor-hookup-guide

4. GrabCAD community offers free CAD files which are ready to build and design for our project. https://grabcad.com/library/hinge-162