EME 154 Padlock Opener Project

Eason Chen

Introduction of the System Modules

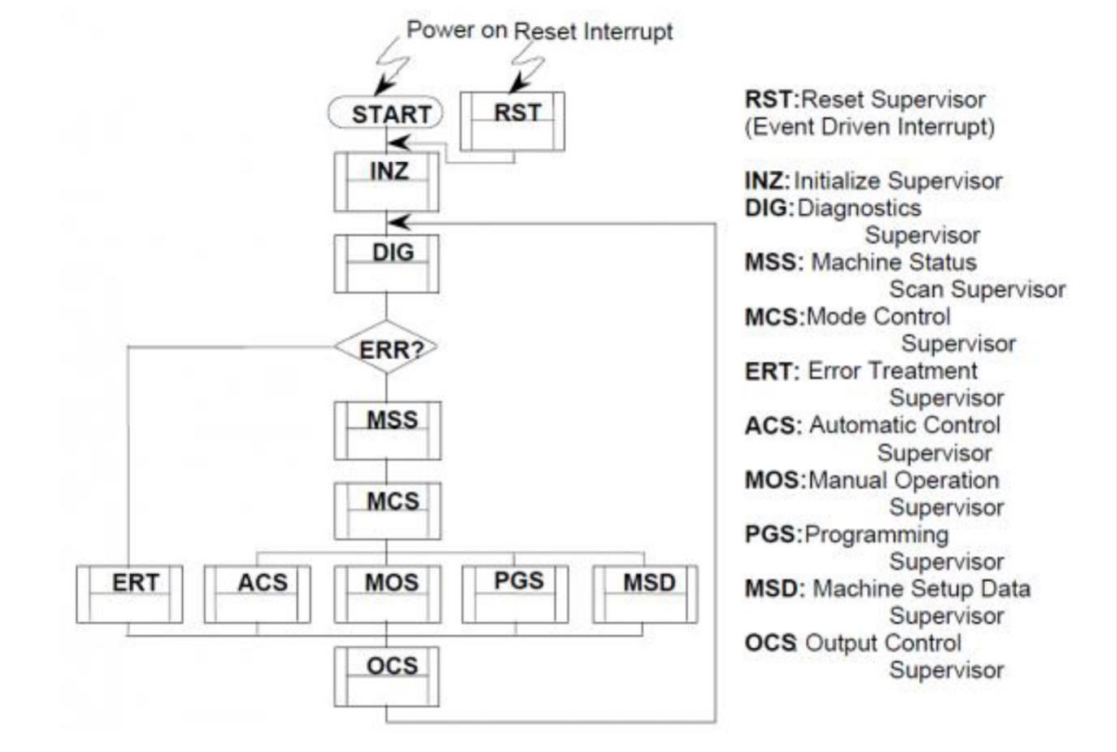
The system modules included below

1. Initialization Function (INZ):
   This is the functions that initialized flags in a system.

2. Diagnostics Supervisor (DIG):
   This is to check the system and detect errors.

3. Machine status selection function (MSS):
   This is used to allow user to input to select the relevant mode.

4. Mode Control Supervisor (MCS):
   Is designed to allow user to select different mode that relate to MSS function.



Introduction of the System Modules

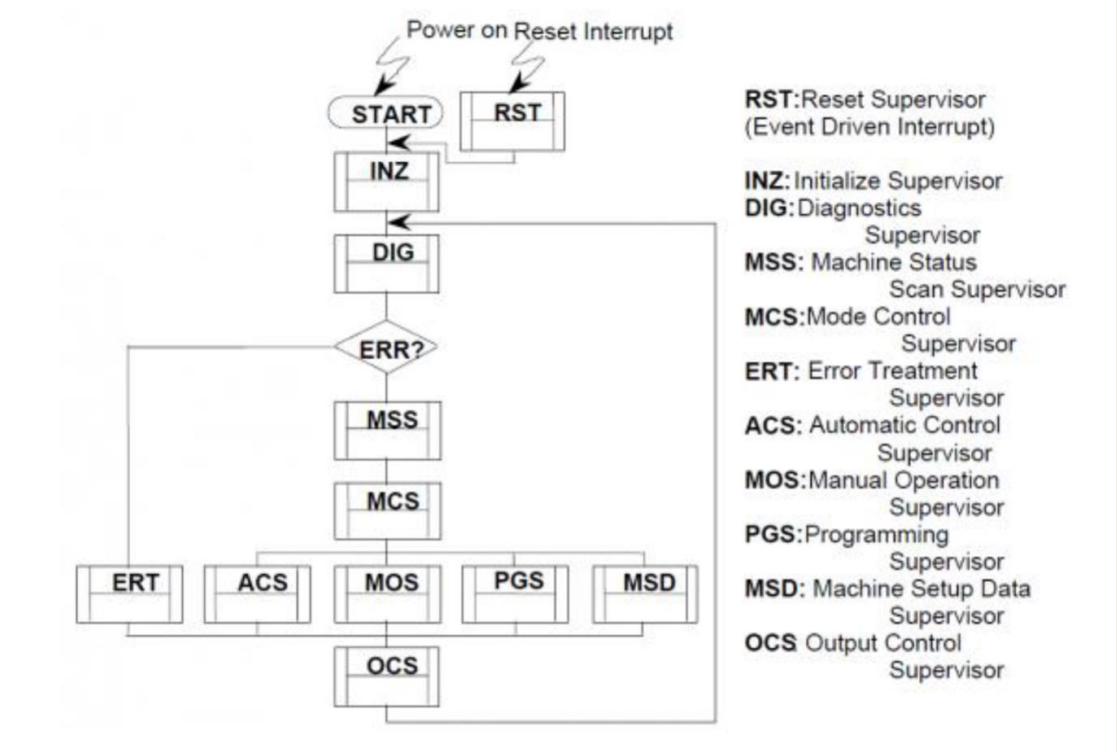
The system modules included below

5. Error Treatment Supervisor (ERT):
   Is to indicate any error, if there is one then push back to initial input.

6. Automatic Control Supervisor (ACS):
   Designed for automatic mode that allow system to open the padlock with the three entries from users.

7. Manual Operation Supervisor (MOS):
   This allow users to control tick by tick on the system manually.

8. Programming Supervisor (PGS):
   NOT APPLIED IN THIS PROJECT.

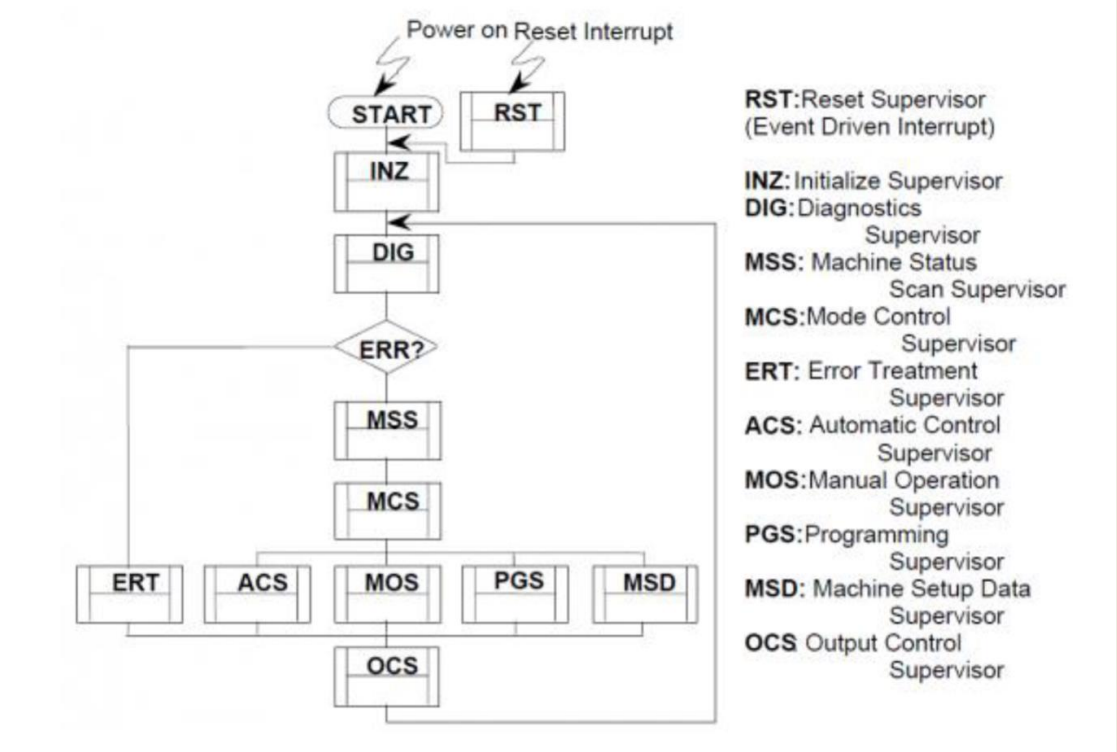


Introduction of the System Modules

The system modules included below

9. Machine Setup Data Supervisor (MSD):
   This is designed to allow users to input their values for automatic control mode.

10. Output Control Supervisor (OCS):
    OCS function is to provide readable information on Serial monitor for users to know which process is ongoing base on the system.



Motor Control Part a.

Evaluation of motor and encoder

- As the motor have different gear ratios, sensitivity to the environment, and the pulses per revolution. By connecting the motor correctly to the microcontroller is provide the best estimation.
- The material that we used for this part are motor shield to provide connection port for motor, Arduino microcontroller, keypad, and a 9V battery.
- With the code provided to evaluate the PPR, we can set up before the P control tuning process.
- In finding the PPR value, we can use the code that control by pressing A or B on keypad for either clockwise or counterclockwise movements. Pulse when reach the initial value and record the data for PPR estimation.

Motor Control Part b.

P-control tuning process of the motor

- In P-control tuning process, by using the provide code we are able to fix the error in motor turning.
- The corresponding Kp value, control loop rate, DIS2GO (the number of pulses in each time period to go). The values in main code are related to the position control as shown in the picture with the function.

```
ISR(TIMER1_COMPA_vect) {
// calculate the position error
if (abs(distanceToGo - encoderPos) < TARGET_DIST){
 positionError = distanceToGo - encoderPos;
}
// positionError = set_distance - encoderPos;
else{
 dist_moved = encoderPos - lastEncoderPos;
 if(distanceToGo > 0){
  positionError += DIS2GO - dist_moved;
 }
 else{
  positionError += -DIS2GO - dist_moved;
 }
}
 // positionError = distanceToGo - encoderPos;
 PWM_value = (int)(KP*(float)positionError);
 // update the last_encoder reading
 lastEncoderPos = encoderPos;
}
```

Motor Control Part b.

P-control tuning process of the motor

- In my tuning process, the Kp value for this motor is 0.12, the Target_Dist is 100, and the DIS2GO value is 5.
- Based on the previous tuning process for PPR value, the number of 813 is used in the calculation to control the motor with a minimum error.

```
#define MAX_PWM 255
#define MIN_PWM 55 // this one depends c
#define PPR 813  // 1080 is ideally, but

// interpolated P-control
#define KP 0.12  // P control parameter
#define TARGET_DIST 100 // pulses
#define DIS2GO 5
```

Human-Machine Interface Design

For the design of Human-Machine interface, I used simple languages to help the users to easily understand the process or ongoing status of the system. In the pictures shown below,

[Fig 1] shows the initial designed automatic operation menu that showing users current mode.

[Fig 2] shows the prompt for users to input value with the error detecting function that shows users to repeat previous step for in-range input. As this example only shows case 1 for entering the first number, similar steps are used for following numbers as well.

```
// set Automatic menu
void setAutomaticMenu(){
  String str1, str2, str3, str4, str5;
  // fill str1~5 to have the Automatic manaul print out correctly
  str1 = "AUTOMATIC OPERATION MENU ";
  str2 = "";
  str3 = "";
  str4 = " ";
  str5 = "5. Exit ";
  str1.toCharArray(menuTitle,25);
  str2.toCharArray(action1, 25);
  str3.toCharArray(action2, 25);
  str4.toCharArray(action3, 25);
  str5.toCharArray(action4, 25);
}
```

Fig 1

```
  Serial.println ("Enter Combination Numbers");

  task=1;

case 1:
  // wait for input of combo1 with function: getTwoCharDigits(number)
  Serial.println ("Enter First Combination Number:");
  combo1 = getTwoCharDigits(number);
  number1 = atoi(number);
  Serial.println (number1);

  if (number1 > 0 && number1 <= 60){
    task = 2;
  }
  else {
    Serial.println ("Input should be from 0-60\n");
    //Serial.println (number1);
```

Fig 2

Human-Machine Interface Design

The Serial.print information shows in below pictures are the following prompt information from previous cases.

[Fig 3] shows the information for users to check the numbers they input and allow them to press # key to give permission for system to process.

[Fig 4] Are the key information for header and menu.

```
case 4:
  //unlock stuff
  //prompt the user to open the padlock by pressing '#'
  Serial.println ("");
  Serial.println ("The Numbers You Entered");
  Serial.print (number1);
  Serial.print (",");
  Serial.print (number2);
  Serial.print (",");
  Serial.print (number3);
  Serial.print (",");
  Serial.println ("");
  // the motor will start moving (lock opening), after the user press '#'
  Serial.println ("Press # to Unlock");
  key = customKeypad.getKey();
   while(key!='#'){
   key = customKeypad.getKey();
   };
```

Fig 3

```
void printHeaderAndMenu(){
  Serial.println("EME-154 Mechatronics");
  Serial.println("Free Time System");
  Serial.println("Eason Chen");
  Serial.println("*****************************************");
  Serial.print("            ");Serial.println(menuTitle);
  Serial.println("*****************************************");
  Serial.println(action1);
  Serial.println(action2);
  Serial.println(action3);
  Serial.println(action4);
  // delay(10); // wait for printing
```

Fig 4

User Manual

- In Machine Set-up mode, which will present as the initial screen and after the execution of the previous operation, users can enter the numbers of the initial position of the pointer relevant to the clock.

- With the correct number entered as the initial position, the system will recognize and prompt users for manual and automatic mode options.

- The motor will move one tick either counterclockwise or clockwise when selecting manual mode according to the input selection.

User Manual

- When selecting automatic mode, the system will execute the following operations:
    - Entering the first combo number.
    - Detection of the error; if an error occurs, prompt message for the new first input value.
    - Entering the second combo number.
    - Detection of the error; if an error occurs, prompt message for the new second input value.
    - Entering the third combo number.
    - Detection of the error; if an error occurs, prompt message for the new third input value.
    - Repeat and show the inputted three numbers in the serial monitor.
    - Motor operated to the input value according to the opening sequence.
    - Prompt user information to open the padlock after the execution.

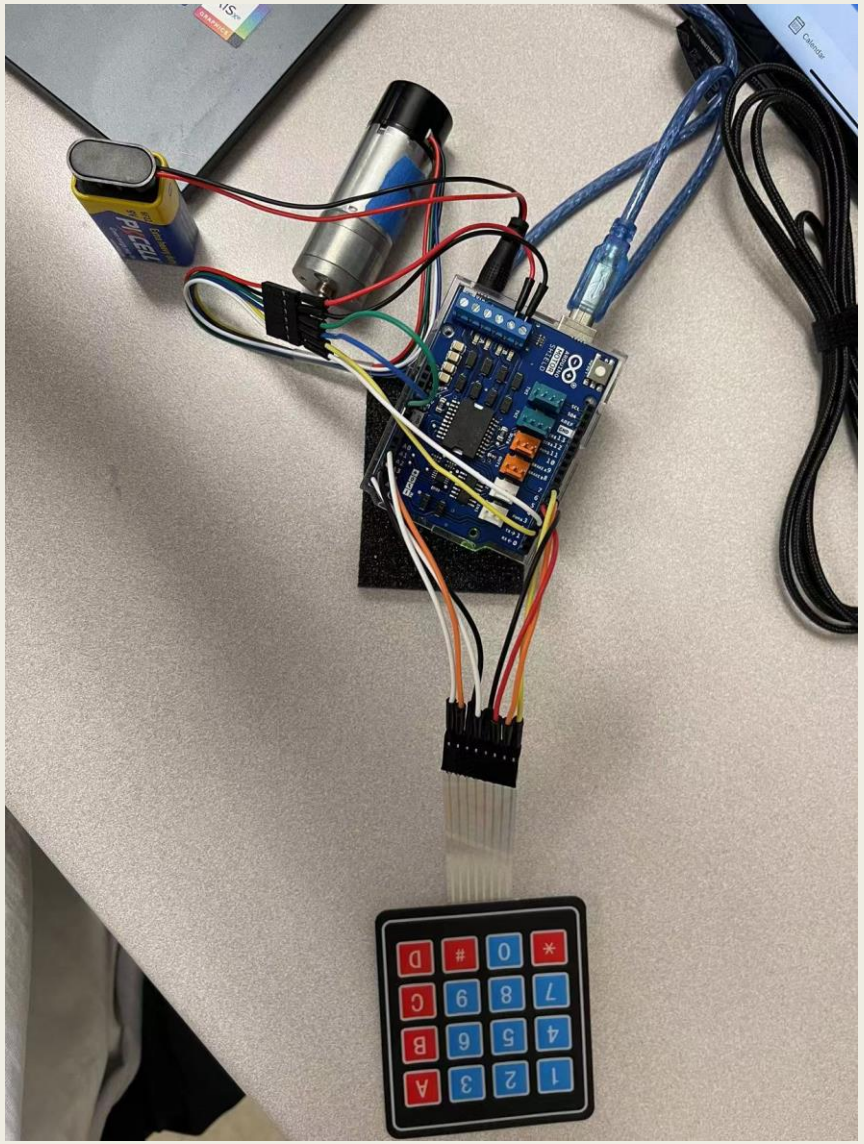
- When pressing the “5” key on the keypad, the system will exit.

Developing Process

- The developing process configured by three parts, as the first one is to built the circuit and run the demo code to check if all the wires and devices are connected correctly. Second part is the complete the code, which mainly focus on the ACS function for automatic control. Third is to check any bugs or errors to make sure the code run correctly and optimize the user interface appearance.

Developing Process

- The first part is to build the circuit as shown in the picture. With correct connection, the motor would be able to move either clockwise or counterclockwise accordingly to the user input.



Developing Process

- Second part is to configure the first three cases in ACS function, that include the first, second, and third number input from users.
- The function also includes the error checking supervisor, and indicator to prompt information to users if the input value not within the suggested range.
- In each of the enter, the serial monitor will also print the current status and input value.

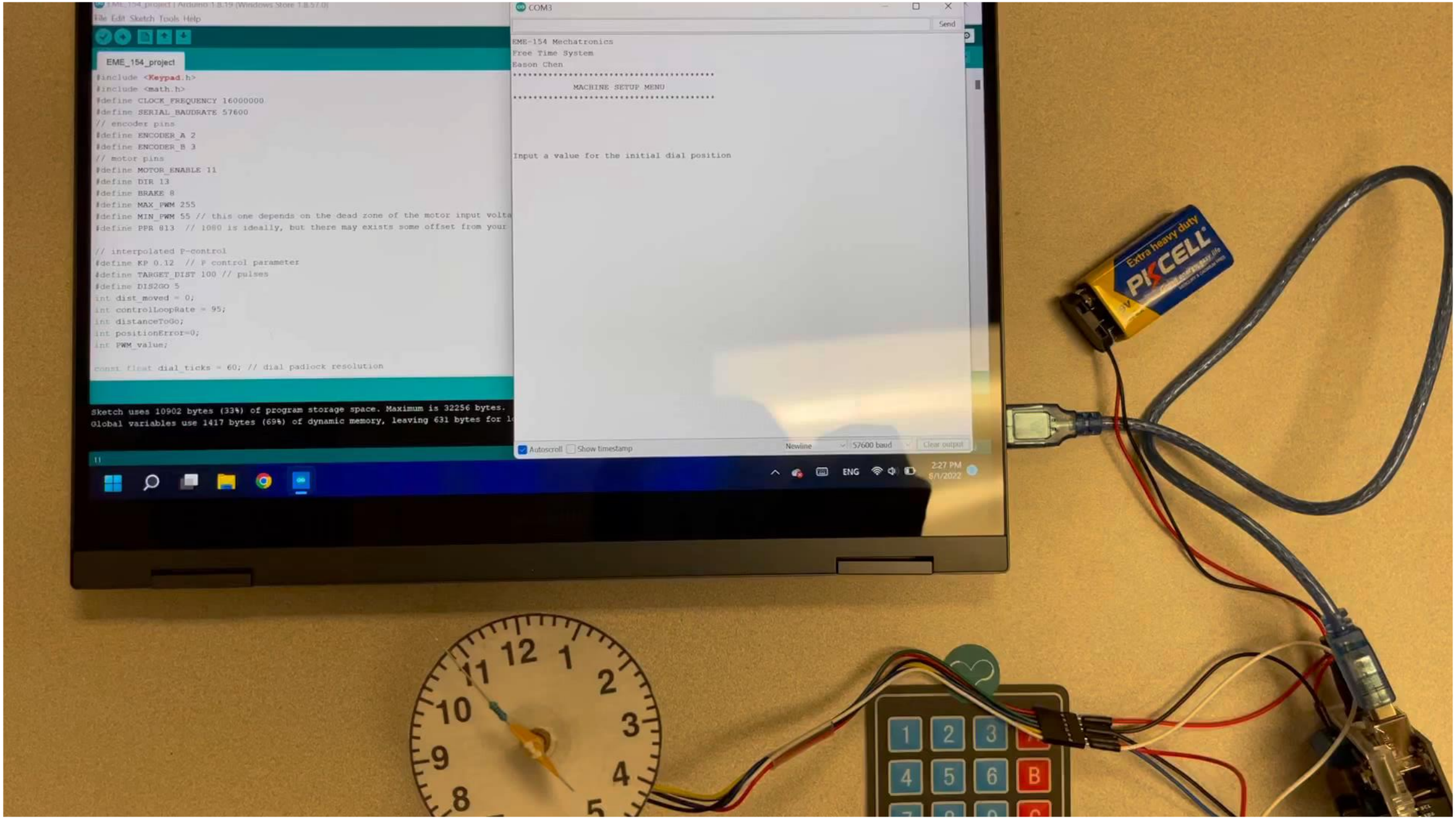
```
case 0:
  // prompt to enter the first combination number combo1 and update machine message
  Serial.println ("Enter Combination Numbers");

  task=1;

case 1:
  // wait for input of combo1 with function: getTwoCharDigits(number)
  Serial.println ("Enter First Combination Number:");
  combo1 = getTwoCharDigits(number);
  number1 = atoi(number);
  Serial.println (number1);

  if (number1 > 0 && number1 <= 60){
    task = 2;
  }
  else {
    Serial.println ("Input should be from 0-60\n");
    //Serial.println (number1);
  }
  break;

  // check the number input if correct or not, reject the number over the range
  // if it over the range, prompt the user to input it again
  // if it is in the range, prompt to input the second number combo2, go to task 2

case 2:
  // wait for input of combo2 with function: getTwoCharDigits(number)
  Serial.println ("Enter Second Combination Number:");
  combo2 = getTwoCharDigits(number);
  number2 = atoi(number);
  Serial.println (number2);
  // check the number input if correct or not, reject the number over the range
  // if it over the range, prompt the user to input it again
  // if it is in the range, prompt to input the third number combo3, go to task 3

  if (number2 > 0 && number2 <= 60){
    task = 3;
  }
  else {
    Serial.println ("Input should be from 0-60\n");
  }
  break;

case 3:
  // wait for input of combo3 with function: getTwoCharDigits(number)
  Serial.println ("Enter Third Combination Number:");
  combo3 = getTwoCharDigits(number);
  number3 = atoi(number);
  Serial.println (number3);
  // check the number input if correct or not, reject the number over the range
  // if it over the range, prompt the user to input it again
  // if it is in the range, go to task 4
```

Developing Process

- In the last case in ACS function, which is the opening sequence and logic for the padlock base on previous user input value.
- The logic include
  - *2 complete turns RIGHT (CW) to the 1st number.*
  - *1 complete turn LEFT (CCW) to the 2nd number.*
  - *Turn RIGHT (CW) to the 3rd number.*
- And final prompt for users pressing # key to return to the main menu.

```
Serial.print (number1);
Serial.print (",");
Serial.print (number2);
Serial.print (",");
Serial.print (number3);
Serial.print (",");
Serial.println ("");
// the motor will start moving (lock opening), after the user press '#'
Serial.println ("Press # to Unlock");
key = customKeypad.getKey();
 while(key!='#'){
 key = customKeypad.getKey();
 };

//int dialPos;
//Move to first location
Serial.print("Moving to: "); Serial.println(combo1);
// move 1 circle CW with function moveCwToDialNO()
moveCwToDialNO(60);
delay(1000);
// move 1 circle CW with function moveCwToDialNO()
moveCwToDialNO(60);
delay(1000);
// move to the 1st number CW with function moveCwToDialNO()
if (combo1 > dial) {
  moveCwToDialNO(combo1-dial);
}
else {
  moveCwToDialNO(combo1-dial+60);
}
//Move to second number
Serial.print("Moving to: "); Serial.println(combo2);
// move 1 circle CCW with function moveCcwToDialNO()
moveCcwToDialNO(-60);
delay(1000);
// move to the 2nd number CCW with function moveCcwToDialNO()

if (combo2 < combo1) {
  moveCwToDialNO(combo2 - combo1);
}
else {
  moveCwToDialNO(-60 + combo2 - combo1);
}
// Move to third number
Serial.print("Moving to: "); Serial.println(combo3);

// move to the 3rd number CW with function moveCwToDialNO()
if (combo3 > combo2) {
  moveCwToDialNO(combo3 - combo2);
}
else {
  moveCwToDialNO(combo3-combo2+60);
}

//pop the lock with the solenoid connected to port 1
Serial.println("You can pull the shackle now!");

// update machine message
Serial.println ("Press # to go back to main menu");
// prompt the user to input '#' to go back the main menu
```

Demo Video


EME_154_project
#include <Keypad.h>
#include <math.h>
#define CLOCK_FREQUENCY 16000000
#define SERIAL_BAUDRATE 57600
// encoder pins
#define ENCODER_A 2
#define ENCODER_B 3
// motor pins
#define MOTOR_ENABLE 11
#define DIR 13
#define BRAKE 8
#define MAX_PWM 255
#define MIN_PWM 55 // this one depends on the dead zone of the motor input volta
#define PPR 813 // 1080 is ideally, but there may exists some offset from your
// interpolated P-control
#define KP 0.12 // P control parameter
#define TARGET_DIST 100 // pulses
#define DIS2GO 5
int dist_moved = 0;
int controlLoopRate = 95;
int distanceToGo;
int positionError=0;
int PWM_value;
const float dial_ticks = 60; // dial padlock resolution
Sketch uses 10902 bytes (33%) of program storage space. Maximum is 32256 bytes.
Global variables use 1417 bytes (69%) of dynamic memory, leaving 631 bytes for l
COM3
Send
EME-154 Mechatronics
Free Time System
Eason Chen
MACHINE SETUP MENU
Input a value for the initial dial position
Autoscroll
Show timestamp
Newline
57600 baud
Clear output
ENG
2:27 PM
Extra heavy duty
PKCELL