Introduction to Robotics and Mechatronics

GROUP 2.5
Benson Chalethu
Pietro Griffa
Ingvar Groza

PreLab 07

Q1

Being:

Nominal step angle = 1.8°

We can easily calculate:

Steps per revolution *= 360°/1.8° = 200*

Q2 & Q3

lab07.ino

```
/*
IRM Lab07 : Closed-loop control I

*/

// Include the required libraries here (already done for you)
#include <string.h>
#include <Adafruit_MotorShield.h>
#include <Wire.h>


// global variables to control each motor
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *myMotor1 = AFMS.getStepper(200, 1);
Adafruit_StepperMotor *myMotor2 = AFMS.getStepper(200, 2);

// pinouts for limit switches
int switch1 = 6; // motor 1
int switch2 = 7; // motor 1
int switch3 = 4; // motor 2
int switch4 = 5; // motor 2

// step size (Number of steps in each iteration)
int stepSize = 20;
// maximum speed in rpm of each motor
int maxSpeed = 500;

// Functions

int move_steps (int steps, int dir, int motor) {
    int switch_check;
    int step_limit, step_stop, step_count, out = 0;
    int ls, switch_press, ref;

  // Check the motor and direction of movement, and set the limiting switch accordingly
    if (motor == 1) {
        if (dir == 1) {
            ls = 7; // limiting switch = switch 2
        }
        else if (dir == 2) {
            ls = 6; // limiting switch = switch 1
        }
        else {
            out = 53;   // if a problem occour in the function, 5 is returned
        }
    }
    else if (motor == 2) {
        if (dir == 1) {
            ls = 5; // limiting switch = switch 4
        }
        else if (dir == 2) {
            ls = 4; // limiting switch = switch 3
        }
        else {
            out = 53;
        }
    }
    else {
        out = 53;
    }
```

Q2 & Q3

```
61
62  // Limit the total number of steps to 999
63  step_limit = 999;
64
65  // Create a loop, which is executed if steps > 0 and the limit switch has not been reached,
66  // in the loop, move the desired motor in small steps (stepsize). Execute the loop until you moved
67  // the whole distance
68  step_count = 0;
69  switch_check = -1;
70  ref = digitalRead(ls);
71
72  if(step_limit <= steps) {
73    step_stop = step_limit;
74  }
75  else {
76    step_stop = steps;
77  }
78
79  /*
80  if (steps != 0 & out == 0) {
81    while (step_count <= step_stop) {
82      if (ls == 7) {
83        myMotor1.step(1, FORWARD, MICROSTEP);
84      }
85      else if (ls == 6) {
86        myMotor1.step(1, BACKWARD,MICROSTEP);
87      }
88      else if (ls == 5) {
89        myMotor2.step(1, FORWARD,MICROSTEP);
90      }
91      else if (ls == 4) {
92        myMotor2.step(1, BACKWARD,MICROSTEP);
93      }
94
95      switch_press = digitalRead(ls);
96      if (switch_press != ref) {
97        switch_check = 0;
98        break;
99      }
100     step_count++
101   }
102 }
103 */
104
105 if (steps > 0 & out == 0) {
106   if (ls == 7) {
107     while (step_count <= step_stop) {
108       myMotor1->step(1, FORWARD, MICROSTEP);
109       switch_press = digitalRead(ls);
110       if (switch_press != ref) {
111         switch_check = 0;
112         break;
113       }
114       step_count++ ;
115     }
116   }
117   else if (ls == 6) {
118     while (step_count <= step_stop) {
119       myMotor1->step(1, BACKWARD,MICROSTEP);
120       switch_press = digitalRead(ls);
121       if (switch_press != ref) {
122         switch_check = 0;
123         break;
124       }
125       step_count++;
126     }
127   }
```

Q2 & Q3

```
    else if (ls == 5) {
      while (step_count <= step_stop) {
        myMotor2->step(1, FORWARD,MICROSTEP);
        switch_press = digitalRead(ls);
        if (switch_press != ref) {
          switch_check = 0;
          break;
        }
        step_count++;
      }
    }
    else if (ls == 4) {
      while (step_count <= step_stop) {
        myMotor2->step(1, BACKWARD,MICROSTEP);
        switch_press = digitalRead(ls);
        if (switch_press != ref) {
          switch_check = 0;
          break;
        }
        step_count++;
      }
    }
  }


  // After running the loop, return ASCII for the switches
    if (switch_check == 0)  {
      switch (ls) {
        // If switch 2 is pressed, return '2'
        case 7 :
          out = 50;
          break;
        // If switch 1 is pressed, return '1'
        case 6 :
          out = 49;
          break;
        // If switch 4 is pressed, return '4'
        case 5 :
          out = 52;
          break;
        case 4 :
        // If switch 3 is pressed, return '3'
          out = 51;
          break;
        default :
          out = 53;
          break;
      }
    }

    // Else, return '0'
    else {
      out = 48;
    }

    return out;   // return the correct message

}
```

Q2 & Q3

```
188  void setup() {
189
190    // Start the serial communication at 9600 baud rate
191    Serial.begin(9600);
192    // Set serial communication timeout to 10
193    Serial.setTimeout(10);
194    // Start the Adafruit Motor Shield and set the maximum speed of the stepper to 500 rpm
195    AFMS.begin();
196    myMotor1->setSpeed(maxSpeed);
197    myMotor2->setSpeed(maxSpeed);
198
199    // Set the input pins
200    pinMode(switch1, INPUT);
201    pinMode(switch2, INPUT);
202    pinMode(switch3, INPUT);
203    pinMode(switch4, INPUT);
204
205  }
206
207
208
209  void loop() {
210    // Initialize parameters
211    char command[50];
212    char check;
213    byte read_check;
214
215    int int_check;
216    int steps, dir, motor;
217    int c4, c3, c2;
218
219    // Check if there is a command on the serial port
220    if (Serial.available() > 0)
221    {
222      read_check = Serial.readBytes(command,20);
223      // If the command is not 5 bytes long, discard it
224      if ((int)read_check == 5)
225      {
226        command[5] = '\0';
227        check = '0';
228      }
229      else
230      {
231        command[0] = '\0';
232        check = '5';
233        Serial.print(check);
234      }
235    }
236
237    // Proper command contains 5 bytes:
238    //  First byte is the stage number: 1 or 2
239    //  Second byte is the direction: 1 or 2
240    //  Third - Fifth bytes are number of steps: 000 - 999
241    if (command[0]!=0)
242    {
243      // Check the first byte and if it is not '1' or '2' discard it
244      // First byte determines the stage (stepper motor) that needs to be moved
245      motor = (command[0]-'0');
246      if (motor != 1 | motor != 2) {
247        check = '5';
248      }
249
```

Q2 & Q3

```
188  void setup() {
189
190    // Start the serial communication at 9600 baud rate
191    Serial.begin(9600);
192    // Set serial communication timeout to 10
193    Serial.setTimeout(10);
194    // Start the Adafruit Motor Shield and set the maximum speed of the stepper to 500 rpm
195    AFMS.begin();
196    myMotor1->setSpeed(maxSpeed);
197    myMotor2->setSpeed(maxSpeed);
198
199    // Set the input pins
200    pinMode(switch1, INPUT);
201    pinMode(switch2, INPUT);
202    pinMode(switch3, INPUT);
203    pinMode(switch4, INPUT);
204
205  }
206
207
208
209  void loop() {
210    // Initialize parameters
211    char command[50];
212    char check;
213    byte read_check;
214
215    int int_check;
216    int steps, dir, motor;
217    int c4, c3, c2;
218
219    // Check if there is a command on the serial port
220    if (Serial.available() > 0)
221    {
222      read_check = Serial.readBytes(command,20);
223      // If the command is not 5 bytes long, discard it
224      if ((int)read_check == 5)
225      {
226        command[5] = '\0';
227        check = '0';
228      }
229      else
230      {
231        command[0] = '\0';
232        check = '5';
233        Serial.print(check);
234      }
235    }
236
237    // Proper command contains 5 bytes:
238    //  First byte is the stage number: 1 or 2
239    //  Second byte is the direction: 1 or 2
240    //  Third - Fifth bytes are number of steps: 000 - 999
241    if (command[0]!=0)
242    {
243      // Check the first byte and if it is not '1' or '2' discard it
244      // First byte determines the stage (stepper motor) that needs to be moved
245      motor = (command[0]-'0');
246      if (motor != 1 | motor != 2) {
247        check = '5';
248      }
249
```

Q2 & Q3

```
250     // Check the second byte and if it is not '1' or '2' discard it
251     // Second byte determines the direction
252     dir = (command[1]-'0');
253     if (dir != 1 | dir != 2) {
254       check = '5';
255     }
256
257     // Check that third to fifth bytes are between '0' and '9'
258     // make sure to convert from chars to integers (subtract 48, the ASCII constant) and multiply accordingly
259     c4 = (command[4]-'0');
260     c3 = (command[3]-'0');
261     c2 = (command[2]-'0');
262     if (c4 >= 0 & c4 <= 9 & c3 >= 0 & c3 <= 9 & c2 >= 0 & c2 <= 9) {
263       steps = c4+10*c3+100*c2;
264     }
265     else {
266       check = '5';
267     }
268     //steps = 1*(command[4]-'0')+10*(command[3]-'0')+100*(command[2]-'0');
269
270     // If everything is fine, move the motors
271     if (check == '5') {
272       // if check == '5' something went wrong
273       // do nothing, it will print out '5'
274     }
275     else {
276       int_check = move_steps (steps, dir, motor);
277       check = int_check;    // divided into two steps to make sure the conversion is made correctly
278     }
279
280     // Check is sent over the serial back to microprocessor:
281     //     '0' if motor moved
282     //     '1' if switch 1 is pressed
283     //     '2' if switch 2 is pressed
284     //     '3' if switch 3 is pressed
285     //     '4' if switch 4 is pressed
286     //     '5' if command is bad
287     Serial.print(check);
288
289     // Reset the command
290     command[0] = '\0';
291     Serial.flush();
292   }
293
294   // Delay in ms
295   delay(5);
296 }
297
```