Jeong Bin Lee
12935084

Mechatronics Assignment 4

Components

Arduino – Arduino UNO

LCD – DFROBOT LCD Keypad Shield V1.1

Pins

LCD

Pins used for LCD are as shown below:

LCD	Arduino Pin
RS	8
Enable	9
d4	4
d5	5
d6	6
d7	7

furthermore, the LCD was placed on top of the Arduino UNO to convenient purposes.

Serial

PrintMessage function is using both \r and \n to end each message.

```
//send a message from arduino to computer using Serial
void PrintMessage(String message)
{
 Serial.print(message);
 Serial.write(13); //carriage return character (ASCII 13, or '\r')
 Serial.write(10); //newline character (ASCII 10, or '\n')
 DelayMilli(200);
}
```

Code

```
#include <LiquidCrystal.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <math.h>

//Author: Jeong Bin Lee
//SID: 12935084
//Date: 30/10/2020
//Status: Finished

/**** C++ ****/
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);


#define START_ROW 4
#define START_COL 2
#define NODE_SIZE 21

/*** ENUM ***/
enum BUTTONS //reads users inputs
{
  NOTHING,
  RIGHT,
  UP,
  DOWN,
  LEFT,
  SELECT
};

/*** GLOBAL VARIABLES ***/
volatile unsigned long int Delay = 0;

uint8_t Button; //enum of user button

uint16_t Adc_Read; //used to read user button

String goal;

//User Position
float row = START_ROW;
float col = START_COL;
//UP = 0, RIGHT = 90, DOWN = 180, LEFT = 270
int16_t face = 90; //initially 90

uint8_t fire_count = 100; //keeps track on where the fire node is

static int main_count = 0; //keeps track on where the current node is
```

```
bool water_goal = false; //true if water goal is achieved, else false

bool fire_goal = false; //true if fire goal is achieved, else false

bool country_road = false; //true if return home is enabled, else false

bool timer = false; //true if select is pressed, else false


//node is the key point the robot visits using the row, col, face, and radius

//each array shows the important variables used in this project

float node_row[NODE_SIZE] = {4.5, 10, 3.5, 3.5, 7, 10.5, 13.5, 12, 8, 15, 17, 18.5, 18.5, 16, 14.5, 11.5, 11.5, 6.5, 4, 4, 7};

float node_col[NODE_SIZE] = {2.5, 2.5, 10, 10, 10, 10, 9, 6.5, 6.5, 2.5, 6.5, 2.5, 11.5, 17.5, 13.5, 13, 17.5, 17.5, 16.5, 14, 14};

int node_face[NODE_SIZE] = {0, 0, 90, 90, 0, 0, 270, 0, 0, 270, 0, 90, 90, 180, 270, 180, 90, 180, 270, 270, 0};

float node_radius[NODE_SIZE] = {3.2, 3.7, 2.7, 2.7, 2.7, 2.2, 2.2, 2.7, 2.2, 2.7, 2.7, 2.2, 5.2, 2.4, 2.7, 2.7, 2.7, 3.2, 3.2, 3.2, 3.2};


//used for timing the project

uint32_t totalsec = 0;

uint8_t mins = 0; //for tellimg number of mins

uint8_t secs = 0; //for telling number of secs


//interrupt service routine

ISR(TIMER2_OVF_vect)

{

 static uint32_t counter = 0;

 static uint32_t millisec = 0;

 Delay++; //Global delay counter (1ms)


 //reading buttons

 if(timer == false)

 {

  counter++;

  if(counter >= 100)

  {

   //TEST//

   ReadUserButton();

   counter = 0;

  }

 }

 else if(timer == true)
```

```
 {
   millisec++;
   if(millisec >= 980)
   {
     totalsec++;
     secs = (totalsec % 3600) % 60;
     mins = (totalsec % 3600) / 60;
     millisec = 0;
     updateTime();
   }
 }
}

//to initialise delay
void DelayInit() {
 TCCR2A = 0;
 // Prescaler for 64
 TCCR2B &= ~(1<<CS20);
 TCCR2B &= ~(1<<CS21);
 TCCR2B |= (1<<CS22);
 TIMSK2 |= (1<<TOIE2); //Enable Overflow interrupt
 TIFR2 |= (1<<TOV2);
}

//to get global delay variable
volatile unsigned long int getDelay()
{
 return Delay;
}

//my delay fuction
void DelayMilli(uint64_t delayTime)
{
 uint64_t count = getDelay();
 while(getDelay() <= (delayTime + count)) {
 }
}

//ADC setup
void ADCSetup(void)
{
 ADCSRA = (1<<ADEN); // Enable ADC
 ADMUX |= (1<<REFS0); // Internal Vcc 5v
}

//read buttons from LCD shield
void ReadUserButton(void)
{
 //read the user input
 if(Adc_Read != ADCsingleREAD(0))
```

```
 {
   Adc_Read = ADCsingleREAD(0);
 }
}

//check what the user pressed
void CheckUserButton(void)
{
 if(Adc_Read <= 50) //Right  = 0
 {
   Serial.println("Right");
   Button = RIGHT;
   DelayMilli(200); //debounce
 }
 else if(600 <= Adc_Read &&
Adc_Read <= 650) //Left   = 626 or
625
 {
   Serial.println("Left");
   Button = LEFT;
   DelayMilli(200); //debounce
 }
 else if(150 <= Adc_Read &&
Adc_Read <= 350) //Up = 208
 {
   Serial.println("Up");
   Button = UP;
   DelayMilli(200); //debounce
 }
 else if(350 <= Adc_Read &&
Adc_Read <= 450)//Down = 410 or
409
 {
   Serial.println("Down");
   Button = DOWN;
   DelayMilli(200); //debounce
 }
 else if(750 <= Adc_Read &&
Adc_Read <= 900)//Select = 824
 {
   Serial.println("Select");
   Button = SELECT;
   DelayMilli(250); //debounce
 }
 else
 {
//   Serial.println("Nothing");
   Button = NOTHING;
//   DelayMilli(150);
 }
}

//read ADC pin once
```

```
int ADCsingleREAD(uint8_t pin)
{
  if(pin == 0)
  {
    ADMUX = 0; //Multiplexer  for which pin to read from
    ADMUX |= (1<<REFS0); // Internal Vcc 5v
  }
  else
  {
    ADMUX = pin;
  }
  ADCSRA |= (1<<ADSC); // start conversion


  // wait for conversion to complete
  while (!(ADCSRA &(1<<ADIF)));

  ADCSRA |= (1<<ADIF);
  return ADC;
}


//send a message from arduino to computer using Serial
void PrintMessage(String message)
{
 Serial.print(message);
 Serial.write(13); //carriage return character (ASCII 13, or '\r')
 Serial.write(10); //newline character (ASCII 10, or '\n')
}


//setup before running the main loop
void setup(void)
{
  Serial.begin(9600);
  Serial.setTimeout(350); //set timeout to 300ms
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("12935084");
  lcd.setCursor(0,1);
  lcd.print("00:00");

  //initialise your code here
  DelayInit();
  ADCSetup();
```

```
 sei(); // enable interrupts
}

//prints the time on LCD shield
void updateTime(void)
{
 lcd.setCursor(0,1);
 if(mins < 10)
 {
  lcd.print("0");
 }
 lcd.print(mins);

 lcd.print(":");

 if(secs < 10)
 {
  lcd.print("0");
 }
 lcd.print(secs);
}

//the main loop where all the code gets executed
void loop()
{
 //MAIN LOOP
 DelayMilli(300);


 CheckUserButton();

 //if select button is pressed
 if(Button == SELECT)
 {
  PrintMessage("CMD_START"); // Start the robot
  DelayMilli(100);
  while(1)
  {
   //moves along the guided path
   if(water_goal == false)
   {
    lcd.setCursor(6,1);
    lcd.print("W"); //finding water
    timer = true; //start timer
    movePath();
   }

   if(goal.toFloat() > 0 && water_goal == false && fire_count != main_count - 1) //find water goal
   {
```

```
      findWater(); //move towards the water goal
      backToNode(); //re-align with the key point
    }
    else if(fire_count < 100 && water_goal == true && fire_goal == false) //back track for fire goal
    {
      lcd.setCursor(6,1);
      lcd.print("F"); //finding fire
      //move back to the fire node
      takeMeHome();
      if((main_count-1) == fire_count)
      {
        findFire(); //find the fire goal
        backToNode();
      }
    }
    else if(water_goal == true && fire_goal == false) //keep going for fire goal
    {
      lcd.setCursor(6,1);
      lcd.print("F"); //finding fire
      //move forward until another goal appears
      movePath();
      if(goal.toFloat() > 0)
      {
        findFire(); //find the fire goal
        backToNode();
      }
    }
    else if(water_goal == true && fire_goal == true) //both goals achieved
    {
      lcd.setCursor(6,1);
      lcd.print("H");
      country_road = true; //return home is true
      takeMeHome();
      DelayMilli(500);
      if(main_count == 0 || (row <= 5.5 && col <= 3.5))
      {
        lcd.setCursor(6,1);
        lcd.print("C");
        PrintMessage("CMD_CLOSE"); //close the bot
        timer = false;
        PrintMessage("CMD_CLOSE"); //close the bot again (in case)
        timer = false;
        while(1);
```

```
      }
    }
    else if((water_goal == false ||
fire_goal == false) && main_count >
20 && fire_goal == 100)
    {
      while(1) //abort mission
      {
        lcd.setCursor(6,1);
        lcd.print("H");
        country_road = true;
        takeMeHome();
        DelayMilli(500);
        if(main_count == 0 || (row <=
5.5 && col <= 3.5))
        {
          lcd.setCursor(6,1);
          lcd.print("C");

PrintMessage("CMD_CLOSE");
          timer = false;

PrintMessage("CMD_CLOSE");
          timer = false;
          while(1);
        }
      }
    }
  }
 }
}

//move back with extreme accuracy
//takes parameter of float for the
distance to move
void moveUp(float dist)
{
 float tmpDist = dist;

 //experimented for maximum
accuracy movement
 if(dist == 0.25)
 {
  dist = (dist/1.04);
 }
 else if(dist == 0.5)
 {
  dist = (dist/1.04) + 0.001;
 }
 else if(dist == 1)
 {
  dist = (dist/1.04) + 0.002;
 }
 else if(dist == 1.5)
```

```
{
  dist = (dist/1.04) + 0.002;
}
else if(dist == 2)
{
  dist = (dist/1.04) + 0.003;
}
else if(dist == 2.5)
{
  dist = (dist/1.04) + (0.05/1.04);
}
else if(dist == 3)
{
  dist = (dist/1.04) + (0.062/1.04);
}
else if(dist == 3.5)
{
  dist = (dist/1.04) + (0.063/1.04);
}
else if(dist == 4)
{
  dist = (dist/1.04) + (0.073/1.04);
}
else if(dist == 4.5)
{
  dist = (dist/1.04) + (0.083/1.04);
}
else if(dist == 5)
{
  dist = (dist/1.04) + (0.094/1.04);
}
else
{
  return;
}
PrintMessage("CMD_ACT_LAT_1_"
+ (String)(dist));

if(face == 0) //UP
{
  row += tmpDist;
}
else if(face == 90) //RIGHT
{
  col += tmpDist;
}
else if(face == 180) //DOWN
{
  row -= tmpDist;
}
else if(face == 270) //LEFT
{
  col -= tmpDist;
```

```
  }


  DelayMilli(400); //delay for the
time it take to move
}


//move back with extreme accuracy
//takes parameter of float for the
distance to move
void moveDown(float dist)
{
  float tmpDist = dist;


  //experimented for maximum
accuracy movement
  if(dist == 0.25)
  {
    dist = (dist/1.04);
  }
  else if(dist == 0.5)
  {
    dist = (dist/1.04) + 0.001;
  }
  else if(dist == 1)
  {
    dist = (dist/1.04) + 0.002;
  }
else if(dist == 1.5)
{
  dist = (dist/1.04) + 0.002;
}
else if(dist == 2)
{
  dist = (dist/1.04) + 0.003;
}
else if(dist == 2.5)
{
  dist = (dist/1.04) + (0.05/1.04);
}
else if(dist == 3)
{
  dist = (dist/1.04) + (0.062/1.04);
}
else if(dist == 3.5)
{
  dist = (dist/1.04) + (0.063/1.04);
}
else if(dist == 4)
{
  dist = (dist/1.04) + (0.073/1.04);
}
else if(dist == 4.5)
{
```

```
  dist = (dist/1.04) + (0.083/1.04);
 }
 else if(dist == 5)
 {
  dist = (dist/1.04) + (0.094/1.04);
 }
 else
 {
  return;
 }

 PrintMessage("CMD_ACT_LAT_0_"
+ (String)(dist));

 if(face == 0) //UP
 {
  row -= tmpDist;
 }
 else if(face == 90) //RIGHT
 {
  col -= tmpDist;
 }
 else if(face == 180) //DOWN
 {
  row += tmpDist;
 }
 else if(face == 270) //LEFT
 {
  col += tmpDist;
 }

  DelayMilli(400); //delay for the
time it take to move
}

//turn with extreme accuracy CCW
//takes parameter angle for the
angle to turn
void turnCCW(int angle)
{
 for(int i=0; i<(angle/0.5); i++)
 {

PrintMessage("CMD_ACT_ROT_0_0.
5");
 }

 if(angle == 90)
 {
  DelayMilli(1000);
 }
 else if(angle == 180)
 {
  DelayMilli(3000);
```

```
  }
  else if(angle > 180)
  {
    DelayMilli(4000);
  }

  face -= angle;

  if(face < 0)
  {
    face = 270;
  }
}

//turn with extreme accuracy CW
//takes parameter angle for the angle to turn
void turnCW(int angle)
{
  for(int i=0; i<(angle/0.5); i++)
  {

PrintMessage("CMD_ACT_ROT_1_0.5");
  }
  if(angle == 90)
  {
    DelayMilli(1000);
  }
  else if(angle == 180)
  {
    DelayMilli(3000);
  }
  else if(angle > 180)
  {
    DelayMilli(4000);
  }

  face += angle;

  if(face > 270)
  {
    face = 0;
  }

}

//scans for any goal near by, ignores the goals outside the radius
//takes parameter float for the radius to scan
//return 0 if outside the radius, or returns the goal scanned within the radius
```

```
String scanGoal(float radius)
{
 String g;
 PrintMessage("CMD_SEN_PING");
 DelayMilli(200);
 g = Serial.readString();
 DelayMilli(100);
 if(g.toFloat() <= radius)
 {
  return g;
 }
 return "0";
}

//scans for any goal near by
//scans the maximum range (5m)
String scanGoal(void)
{
 String g;
 PrintMessage("CMD_SEN_PING");
 DelayMilli(100);
 g = Serial.readString();
 DelayMilli(100);

 return g;
}

//approach the water goal until achieved
void findWater(void)
{
 String g1, g2;
 String goalID;

 PrintMessage("CMD_SEN_GOAL");
 DelayMilli(500);
 goalID = Serial.readString();
 DelayMilli(500);
 if(goalID.toFloat() == 1)
 {
  water_goal = true;
  return;
 }

 do
 {
  g1 = scanGoal();
  moveUp(0.25);
  DelayMilli(500);
  g2 = scanGoal();

  if(g1.toFloat() == 0)
  {
```

```
    g1 = "100";
   }
   if(g2.toFloat() == 0)
   {
    g2 = "100";
   }

   if(g1.toFloat() > g2.toFloat())
   {
    while(g1.toFloat() > g2.toFloat())
    {
     g1 = g2;
     moveUp(0.25);
     DelayMilli(500);
     g2 = scanGoal();

     //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
     DelayMilli(500);
     goalID = Serial.readString();
     DelayMilli(500);
     if(goalID.toFloat() == 1)
     {
      water_goal = true;
      return;
     }

     //check if the goal can be ID
     if(g2.toFloat() < 2.0)
     {

PrintMessage("CMD_SEN_ID");
      DelayMilli(100);
      String ID = Serial.readString();
      if(ID.toFloat() == 2)
      {
       lcd.print("E1");
       fire_count = main_count - 1;
//store the case in main loop
       return;
      }
     }
     else if(g2.toFloat() == 0)
     {
      break;
     }
    }
    moveDown(0.25);
    DelayMilli(500);
   }
   else if (g2.toFloat() > g1.toFloat())
   {
    moveDown(0.25);
```

```
    DelayMilli(500);
    while(g2.toFloat() > g1.toFloat())
    {
     g2 = scanGoal();
     moveDown(0.25);
     DelayMilli(500);
     g1 = scanGoal();

     //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
     DelayMilli(500);
     goalID = Serial.readString();
     DelayMilli(500);
     if(goalID.toFloat() == 1)
     {
      water_goal = true;
      return;
     }

     //check if the goal can be ID
     if(g1.toFloat() < 2.0)
     {

PrintMessage("CMD_SEN_ID");
       String ID = Serial.readString();
       if(ID.toFloat() == 2)
       {
        fire_count = main_count - 1;
//store the case in main loop
         return;
       }
      }
      else if(g1.toFloat() == 0)
      {
       break;
      }
     }
     moveUp(0.25);
     DelayMilli(500);
    }

    //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
    DelayMilli(500);
    goalID = Serial.readString();
    DelayMilli(500);
    if(goalID.toFloat() == 1)
    {
     water_goal = true;
     return;
    }
```

```
  turnCW(90);
  DelayMilli(2000);

  g1 = scanGoal();
  moveUp(0.25);
  DelayMilli(500);
  g2 = scanGoal();

  if(g1.toFloat() > g2.toFloat())
  {
   while(g1.toFloat() > g2.toFloat())
   {
    g1 = g2;
    moveUp(0.25);
    DelayMilli(500);
    g2 = scanGoal();

    //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
    DelayMilli(500);
    goalID = Serial.readString();
    DelayMilli(500);
    if(goalID.toFloat() == 1)
    {
     water_goal = true;
     return;
    }

    //check if the goal can be ID
    if(g2.toFloat() < 2.0)
    {

PrintMessage("CMD_SEN_ID");
     String ID = Serial.readString();
     if(ID.toFloat() == 2)
     {
      fire_count = main_count - 1;
//store the case in main loop
      return;
     }
    }
   }
   moveDown(0.25);
   DelayMilli(500);
  }
  else if (g2.toFloat() > g1.toFloat())
  {
   moveDown(0.25);
   DelayMilli(500);
   while(g2.toFloat() > g1.toFloat())
   {
    g2 = scanGoal();
    moveDown(0.25);
```

```
      DelayMilli(500);
      g1 = scanGoal();

      //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
      DelayMilli(500);
      goalID = Serial.readString();
      DelayMilli(500);
      if(goalID.toFloat() == 1)
      {
       water_goal = true;
       return;
      }

      //check if the goal can be ID
      if(g1.toFloat() < 2.0)
      {

PrintMessage("CMD_SEN_ID");
        String ID = Serial.readString();
        if(ID.toFloat() == 2)
        {
          fire_count = main_count - 1;
//store the case in main loop
          return;
        }
          }
         }
          moveUp(0.25);
        }

        //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
        DelayMilli(500);
        goalID = Serial.readString();
        DelayMilli(500);
        if(goalID.toFloat() == 1)
        {
          water_goal = true;
          return;
        }
       } while(goalID.toFloat() != 1);
}


//takes the robot back to the
current node
void backToNode(void)
{
 float minDist = 100, dist;
 float r, c;
```

```
 //go back to the node
 if(main_count <= 0)
 {
  r = node_row[0];
  c = node_col[0];
 }
 else
 {
  r = node_row[main_count - 1];
  c = node_col[main_count - 1];
 }

 //face UP
//  turnCCW(face);
 while(r != row || c != col)
 {
  if(face == 0)
  {
   while(r != row)
   {
    if(r > row)
    {
     moveUp(0.25);
    }
    else if(r < row)
    {
     moveDown(0.25);
    }
    DelayMilli(200);
   }
  }
  else if(face == 180)
  {
   while(r != row)
   {
    if(r > row)
    {
     moveDown(0.25);
    }
    else if(r < row)
    {
     moveUp(0.25);
    }
    DelayMilli(200);
   }
  }
  //face RIGHT
//  turnCW(abs(90 - face));
  else if(face == 90)
  {
   while(c != col)
   {
```

```
    if(c > col)
    {
      moveUp(0.25);
    }
    else if(c < col)
    {
      moveDown(0.25);
    }
    DelayMilli(200);
  }
 }
 else if(face == 270)
 {
  while(c != col)
  {
    if(c > col)
    {
      moveDown(0.25);
    }
    else if(c < col)
    {
      moveUp(0.25);
    }
    DelayMilli(200);
  }
 }
```

```
  //turn 90 and do it again if row or col doesnt match
  if(r != row || c != col)
  {
    turnCW(90);
  }
 }

 //face the correct direction
 if((node_face[main_count - 1] - face) > 0)
 {

turnCW(abs(node_face[main_count - 1] - face));
   DelayMilli(2000);
 }
 else if((node_face[main_count - 1] - face) < 0)
 {

turnCCW(abs(node_face[main_count - 1] - face));
   DelayMilli(2000);
 }

 //find the distance to each node
 //find the smallest distance for the nodes
```

```
  //find the difference in row && move 0.5m closer until row is reached
  //find the difference in col && move 0.5m closer until col is reached
}


//approach the fire goal until achieved
void findFire(void)
{
  //if the current node is equal to fire node
  String g1, g2;
  String goalID;

  //check if goal is achieved
  PrintMessage("CMD_SEN_GOAL");
  DelayMilli(500);
  goalID = Serial.readString();
  DelayMilli(500);
  if(goalID.toFloat() == 2)
  {
    fire_goal = true;
    return;
  }

do
{
  g1 = scanGoal();
  moveUp(0.25);
  DelayMilli(500);
  g2 = scanGoal();

  if(g1.toFloat() == 0)
  {
    g1 = "100";
  }
  if(g2.toFloat() == 0)
  {
    g2 = "100";
  }

  if(g1.toFloat() > g2.toFloat())
  {
    while(g1.toFloat() > g2.toFloat())
    {
      g1 = g2;
      moveUp(0.25);
      DelayMilli(500);
      g2 = scanGoal();
      //check if goal is achieved
```

```
PrintMessage("CMD_SEN_GOAL");
      DelayMilli(500);
      goalID = Serial.readString();
      DelayMilli(500);
      if(goalID.toFloat() == 2)
      {
       fire_goal = true;
       return;
      }
     }
     moveDown(0.25);
   }
   else if (g2.toFloat() > g1.toFloat())
   {
    moveDown(0.25);
    while(g2.toFloat() > g1.toFloat())
    {
     g2 = scanGoal();
     moveDown(0.25);
     DelayMilli(500);
     g1 = scanGoal();
     //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
      DelayMilli(500);
      goalID = Serial.readString();
      DelayMilli(500);
      if(goalID.toFloat() == 2)
      {
       fire_goal = true;
       return;
      }
     }
     moveUp(0.25);
     DelayMilli(500);
   }

   //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
   DelayMilli(500);
   goalID = Serial.readString();
   DelayMilli(500);
   if(goalID.toFloat() == 2)
   {
    fire_goal = true;
    return;
   }

   turnCW(90);
   DelayMilli(2000);

   g1 = scanGoal();
```

```
  moveUp(0.25);
  DelayMilli(500);
  g2 = scanGoal();

  if(g1.toFloat() > g2.toFloat())
  {
   while(g1.toFloat() > g2.toFloat())
   {
    g1 = g2;
    moveUp(0.25);
    DelayMilli(500);
    g2 = scanGoal();
    //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
    DelayMilli(500);
    goalID = Serial.readString();
    DelayMilli(500);
    if(goalID.toFloat() == 2)
    {
     fire_goal = true;
     return;
    }
   }
   moveDown(0.25);
   DelayMilli(500);
  }
  else if (g2.toFloat() > g1.toFloat())
  {
   moveDown(0.25);
   DelayMilli(500);
   while(g2.toFloat() > g1.toFloat())
   {
    g2 = scanGoal();
    moveDown(0.25);
    DelayMilli(500);
    g1 = scanGoal();
    //check if goal is achieved

PrintMessage("CMD_SEN_GOAL");
    DelayMilli(500);
    goalID = Serial.readString();
    DelayMilli(500);
    if(goalID.toFloat() == 2)
    {
     fire_goal = true;
     return;
    }
   }
   moveUp(0.25);
   DelayMilli(500);
  }

  //check if goal is achieved
```

```
PrintMessage("CMD_SEN_GOAL");
   DelayMilli(500);
   goalID = Serial.readString();
   DelayMilli(500);
   if(goalID.toFloat() == 2)
   {
    fire_goal = true;
    return;
   }
 } while(goalID.toFloat() != 2);
}



//move through the main path and scan for goals
void movePath(void)
{
 switch(main_count)
 {
   DelayMilli(500);
   case 0:
    moveUp(0.5); //1
    turnCCW(90); //2
    moveUp(0.5); //3
    //4
    break;
case 1:
 moveUp(1.5); //5
 moveUp(4); //6
 //7
 break;
case 2:
 moveDown(1.5); //8
 moveDown(5); //9
 turnCW(90); //10
 moveUp(4); //11
 //12
 break;
case 3:
 moveUp(3.5); //13
 DelayMilli(500);
 //14
 break;
case 4:
 turnCCW(90); //15
 moveUp(3.5); //16
 //17
 break;
case 5:
 moveUp(3.5); //18
 //19
 break;
```

```
  case 6:
    moveUp(3); //20
    turnCCW(90); //21
    moveUp(1); //22
    //23
    break;
  case 7:
    moveUp(2.5); //24
    turnCW(90); //25
    moveDown(1.5); //26
    //27
    break;
  case 8:
    moveDown(4); //28
    //29
    break;
  case 9:
    moveUp(5); //30
    moveUp(2); //31
    turnCCW(90); //32
    moveUp(4); //33
    //34
    break;
  case 10:
    moveDown(4); //35
    turnCW(90); //36
    moveUp(2); //37
    //38
    break;
  case 11:
    moveUp(1.5); //39
    turnCW(90); //40
    moveDown(4); //41
    //42
    break;
  case 12:
    moveUp(4.5); //43
    moveUp(4.5); //44
    //45
    break;
  case 13:
    moveUp(5); //46
    moveUp(1); //47
    turnCW(90); //48
    moveUp(2.5); //49
    //50
    break;
  case 14:
    moveUp(1.5); //51
    turnCW(90); //52
    moveUp(4.5); //53
    //54
```

```
    break;
  case 15:
    turnCCW(90); //55
    moveUp(3); //56
    //57
    break;
  case 16:
    turnCCW(90); //58
    moveUp(4.5); //59
    //60
    break;
  case 17:
    turnCW(90); //61
    moveUp(5); //62
    //63
    break;
  case 18:
    moveUp(2.5); //64
    turnCW(90); //65
    moveUp(1); //66
    //67
    break;
  case 19:
    moveUp(2.5); //68
    //69
    break;
  case 20:
    turnCW(90); //70
    moveUp(3); //71
    //72
    break;
  }

  if(main_count > 20)
  {
    main_count = 20;
  }

  DelayMilli(1000);
  goal =
scanGoal(node_radius[main_count])
;
  if(goal.toFloat() == 0)
  {
    DelayMilli(250);
    goal =
scanGoal(node_radius[main_count])
;
    DelayMilli(250);
  }


  main_count++;
```

```
  //check if goal is achieved
  PrintMessage("CMD_SEN_ID");
  DelayMilli(500);
  String goalID = Serial.readString();
  DelayMilli(500);
  if(goalID.toFloat() == 2)
  {
    fire_count = main_count - 1;
  }
}



//back track to starting position by going back its original path
void takeMeHome(void)
{
  if(main_count > 21)
  {
    main_count = 21;
  }
  main_count--;
  switch(main_count)
  {
    case 20:
      moveDown(3);
      turnCCW(90);
      break;
    case 19:
      moveDown(2.5);
      break;
    case 18:
      moveDown(1);
      turnCCW(90);
      moveDown(2.5);
      break;
    case 17:
      moveDown(5);
      turnCCW(90);
      break;
    case 16:
      moveDown(4.5);
      turnCW(90);
      break;
    case 15:
      moveDown(3);
      if(country_road == false)
      {
        turnCW(90);
      }
      break;
    case 14:
      if(country_road == true ||
(fire_count != 13))
```

```
    {
      moveDown(4);
      turnCCW(90);
      moveDown(1.5);
      main_count--;
    }
    else
    {
      moveDown(4.5);
      turnCCW(90);
      moveDown(1.5);
    }
    break;
  case 13:
    moveDown(2.5);
    turnCCW(90);
    moveDown(1);
    moveDown(5);
    break;
  case 12:
    if(country_road == true ||
(fire_count != 11))
    {
      moveDown(5);
      turnCCW(90);
      moveDown(1.5);
      main_count--;
    }
    else
    {
      moveDown(4.5);
      moveDown(4.5);
    }
    break;
  case 11:
    moveUp(4);
    turnCCW(90);
    moveDown(1.5);
    break;
  case 10:
    if(country_road == true ||
(fire_count != 9))
    {
      moveDown(5);
      moveDown(4);
      main_count--;
    }
    else if(fire_count <= 6)
    {
      moveDown(3.5);
      turnCCW(90);
      moveDown(2.5);
      main_count = 7;
    }
```

```
    else
    {
      moveDown(2);
      turnCCW(90);
      moveUp(4);
    }
    break;
   case 9:
    moveDown(4);
    turnCW(90);
    moveDown(2);
    moveDown(5);
    break;
   case 8:
    if(country_road == true ||
(fire_count < 1 && fire_count > 8))
    {
      turnCW(90);
      moveDown(4);
      turnCCW(90);
      moveDown(3.5);
      main_count = 1;
    }
    else
    {
      moveUp(4);
    }
    break;
  case 7:
    moveUp(1.5);
    turnCCW(90);
    moveDown(2.5);
    break;
  case 6:
    moveDown(1);
    turnCW(90);
    moveDown(3);
    break;
  case 5:
    moveDown(3.5);
    break;
  case 4:
    moveDown(3.5);
    turnCW(90);
    break;
  case 3:
    moveDown(3.5);
    break;
  case 2:
    if(country_road == true )
    {
      moveDown(4);
      main_count = 1;
```

```
    }
    else if(fire_count != 1)
    {
      moveDown(4);
      turnCCW(90);
      moveUp(1);
      main_count = 1;
    }
    else
    {
      moveDown(4);
      turnCCW(90);
      moveUp(5);
      moveUp(1.5);
    }
    break;
  case 1:
    moveDown(4);
    moveDown(1.5);
    break;
  case 0:
    moveDown(0.5);
    turnCW(90);
    moveDown(0.5);
    break;
}
  DelayMilli(2000);
}
```