June 18, 2019

Spazzy Jazzy

Jessica Zhang, Angela Zhang

Mr. Robert Webb

Jessica's Reflection

Though I was only able to take computer tech twice during high school, it's still one of my favourite courses. I really like the atmosphere of the classroom. There's the people who slack off and the people who work, but there's always conversation and laughter in the classroom. Troubleshooting, though sometimes frustrating, is also extremely gratifying when it works out. Everyone's really collaborative and willing to help.

One of the things that make this course so great is Mr. Webb. He gives us the room to make our own mistakes, but also our own successes (though he's always down to help you, if you need it). With this freedom, it's easy to see that we get out of this project exactly what we put in. I'm sure there are teachers who wouldn't believe high school students would be able to create a bot that navigates a maze and puts out a fire under minimal guidance/supervision. Mr. Webb knows that we can, though, and sees the results firsthand. He pushes us to this high level and never believes we're incapable of reaching it. He's also good at humbling us when we start getting too cocky.

For me, I wish I had the chance to take this class in Grade 11 too; there's a bunch of stuff I would have learned making the sumobot that would have been really helpful to making the firefighter. I also would have liked a page of rules and regulations my bot had to follow (see Misc Tips, #3) because there was some stuff I didn't know not to do until I did it. I'm still very happy with the way our bot turned out and I'm glad I took this class before I went to University. Tech is the BEST.

Tips for future grade 12 students / things I wish I knew heading in:
MISCELLANEOUS

- On the first day of class, Webb spits out a BUNCH of useful information quickly and conversationally - get out a pencil and paper and write down everything you can catch
- buy/order any auxiliary materials as soon as possible (eg. fan, jumper wires, metal rods for your bot body, candles and lighter towards the end of the course)

- If you did not take the Grade 11 course, ask Webb to list out all the rules your bot must follow (fan cannot turn on until it reaches within 30 cm of the flame, bot cannot touch the flame block, etc)
- Get the deadlines down (they tend to creep up on you) and a functional, working bot as soon as possible. The extra time to mess with the code in the end is a lot of help
- If something's not working, try to figure it out yourself first. Then ask your partner. Then ask another group. Then ask the stacked group that's way ahead of everyone else and is really good at tech. And then, if no one has ANY clue on how to answer your question/solve your problem, you can ask Webb
- The multimeter is your best friend

PCB

- GROUNDS BETWEEN YOUR BOARDS MUST CONNECT !!!!!!
- Design all your components on one PCB if you can. Messing around with wires sucks
- Have all your power supplies going into one place on your PCB, and then use wires to connect them board to board
- Double, triple, quadruple check your PCB design and your components orientation. FIXING THIS IS HARD if you mess up
- When designing the PCB, solder in some headers to connect to the "unused" ports on your PIC. These come in use later when you're troubleshooting

PROGRAMMING

- One option instead of Microcode Studio is Great Cow Basic. Advantages include being able to download to your laptop so you can code/program at home, simpler syntax
- Make a new file for every day you're working on your code, and copy everything from yesterday into the new file. You'll appreciate the backups once you change

a bunch of things and they don't work at all. Alternatively, (if you're on your laptop with Great Cow Basic) github

- Comment your code. I could barely remember what the code I wrote myself was supposed to do, never mind my partners code

Bot Design

- Wheels in the center of the sides of the bot allows the bot to pivot in place
- Putting the wall detection sensors NOT over the wheels makes a smoother bot
- It helps during troubleshooting to make the motherboard easily accessible
- Omniwheels look slick, move fast, and are easier to program maze-navigating algorithms for, but also expensive. Regular wheels and turning also works just fine
- Don't spend too much time arguing the advantages/disadvantages of every single design choice - it's okay to make decisions
- Circular bots, or bots with rounded corners got caught/stuck less when navigating turns

Angela's Reflection

Through my two years in tech, the most important thing I have learned is how to troubleshoot hardware-related problems and channel frustration in a healthy way. Tech has taught me to approach problems more effectively, carrying into other aspects of my life besides the course itself. After taking a few years of tech, problem solving starts to become second nature — when you encounter problems, you start thinking of solutions and alternative methods to get things done. Even if you don't end up in a university program related to engineering, this kind of mindset is good to have.

In the beginning of grade 11 tech, I had no idea how a schematic worked, what Traxmaker was, and what a PCB is supposed to do. Through building the LED cube, I was familiarized with the typical workflow required for a tech project: reading a schematic, using Traxmaker to create a circuit diagram, creating transparencies, etching & drilling, placing components and soldering.

Generally you want to be independent and troubleshoot on your own as much as you can. However, if you're not sure that you have the right idea, ask. I found this helped me from going down the wrong track and wasting my time and sanity.

Thank you Mr. Webb, for having the patience to stay after school to supervise me in the tech room on numerous occasions in grade 11. I really appreciate the help that you provide us, especially in teaching us about the things that are relevant to our future but that not many people know of, such as becoming an electrician without going to university and what to do if your car crashes into a utility pole.

Tips for the Firefighter:

- If you are trying to achieve modularity, plan this ahead well. Make sure you know exactly which wires are connected from board to board so that you can decide where to use terminal blocks.
- A lot of students in the class chose to use Great Cow Basic because you can download the program on your personal computer and take the chip home to

program. There is also reliable documentation online. However Great Cow only has nice UI for Windows.

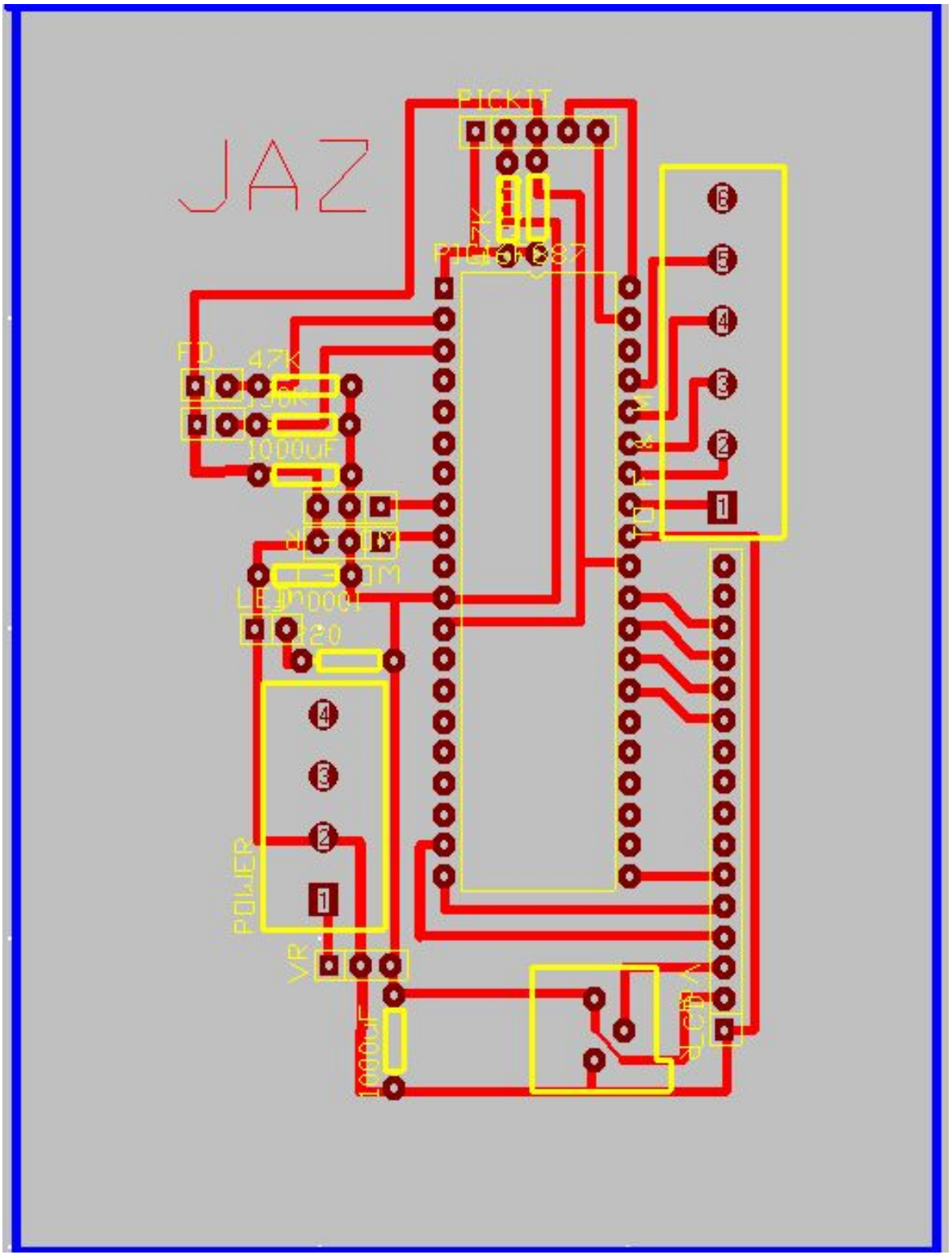
- For the sumo-bot, we had a hard time meeting our deadlines. Once we missed one deadline, it was hard meeting the other deadlines because it's a positive feedback loop. Don't miss any deadlines if you can.
- Also don't fall prey to the midterm crunch and subsequent onset of senioritis. Although marks will no longer motivate you, don't let this be a reason to not meet deadlines because eventually when you do get to the maze, time really counts.
- You will need female-to-female and male-to-female jumper wires. Most people order them off Amazon.
- Sometimes unexpected problems arise. For us, our fan would only work if we coded using binary. Another group had a broken wire in the ribbon cable that wasn't obvious at all. We also spent days troubleshooting when it was a faulty potentiometer — we thought it was either our wiring or our program, and we redid both several times. It is on these stupid problems that you don't want to waste time on. Try everything because *nothing is guaranteed to be not be the problem*.
- Hot glue should be used only where it can't be seen. You want your bot to look professional for interviews.
- Left wall hugging works better for the mazes.
- Have one person in your class take the order to buy fans.
- Use velcro to stick the fan to your bot for removal later on.
- Make sure the hollow part of the fan is facing up so that it can draw in air.
- Plan out the length of your ribbon wire accordingly so that you can reach all parts of the maze.
- Some groups chose to use omni wheels. They are more expensive and require more ports. You should try to optimize this expensive purchase by having your program save on time and distance travelled to blow out the flame.
- Your bot needs to be less than 30 cm from the candle but it also can't touch the wood blocks that holds the candle up to blow the flame out.

- Your bot has to be high enough to reach the height of the candle.
- The distance sensor shouldn't be over the wheels.

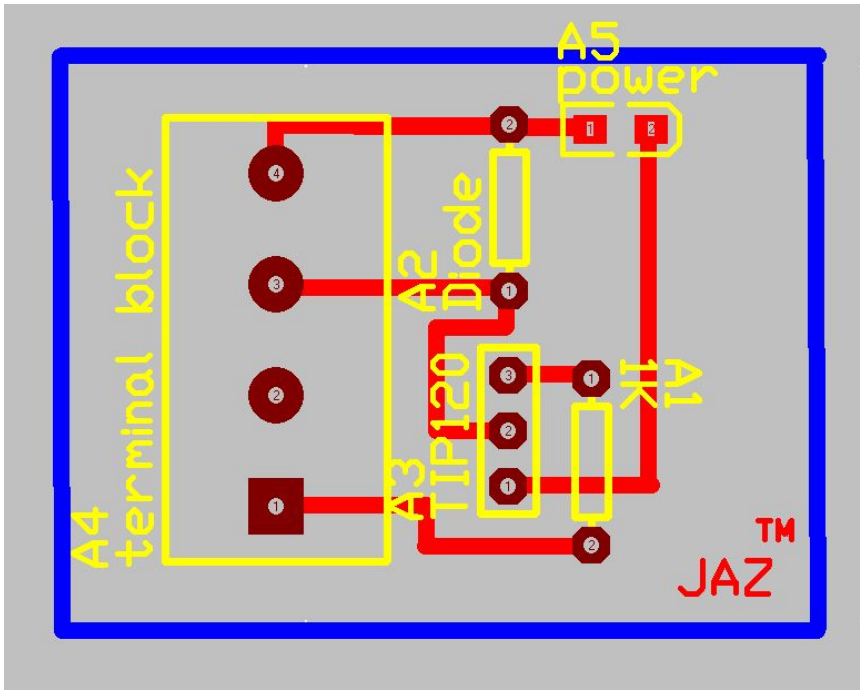
What I liked about TEJ4MI was having Jessica Zhang as my partner. I liked that we were able to joke around and have conversations but also get work done. What can be done to improve the Grade 11 course is to help students new to tech adjust.

My suggestion is to pair new students with students in the class who are more experienced. An incentive for the more experienced student is to offer them bonus marks.

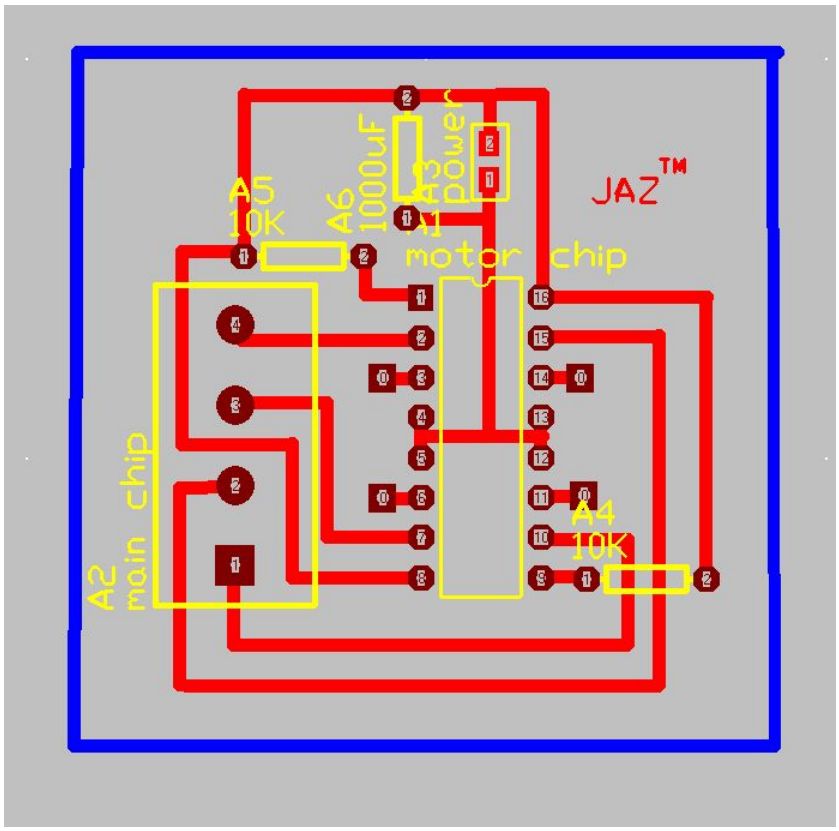
Circuit Boards



Motherboard



Fan Board



Motor Board

Program for Maze 1 [COMPLETE]

```
  'CHIP SETUP
#chip 16f887, 4
#config OSC = INTOSCIO

'LCD SETUP
#define LCD_IO_ 4
#define LCD_RS  PORTD.0
#define LCD_RW  PORTD.1
#define LCD_Enable PORTD.2
#define LCD_DB4 PORTD.4
#define LCD_DB5 PORTD.5
#define LCD_DB6 PORTD.6
#define LCD_DB7 PORTD.7
dir PORTD out

'MOTORS AND FAN SETUP
#define RMB PORTB.0
#define RMF PORTB.1
#define LMB PORTB.2
#define LMF PORTB.3
#define FAN PORTB.4
dir PORTB out

'SENSORS SETUP
#define WDF an5
#define WDR an6
#define FD an1
#define LD an0
dir PORTA in
dir PORTE in

'VARIABLES SETUP
dim FDVal as Byte
dim WDFVal as Byte
dim WDLVal as Byte
dim LDVal as Bit
dim LDCount as Byte
LDCount = 0
dim lineAlreadyDetected as Bit
lineAlreadyDetected = 0
dim dist as Byte
```

```
dist = 22

function flameDetected
  'returns 0 if the flame is not detected, 1 if the flame is
detected
    if FDVal > 8 then
      flameDetected = 0
    else
      flameDetected = 1
    end if
end Function

sub loadSensorVal
  'SENSORS INPUT
  FDVal = ReadAD(FD)
  WDLVal = (((6787/(ReadAD(WDR)-3)))-4)/5
  WDFVal = (((6787/(ReadAD(WDF)-3)))-4)/5

  if ReadAD(LD) > 50 Then
    LDVal = 1
  else
    LDVal = 0
  end if

end sub

sub moveF
  set LMF on
  set RMF on
  set LMB off
  set RMB off
end sub

sub moveB
  set LMF off
  set RMF off
  set LMB on
  set RMB on
end sub

sub turnL
  set LMF off
  set RMF on
  set LMB off
```

```
  set RMB off
end sub

sub turnR
  set LMF on
  set RMF off
  set LMB off
  set RMB off
end sub

sub pivotL
  set LMF off
  set RMF on
  set LMB on
  set RMB off
end sub

sub pivotR
  set LMF on
  set RMF off
  set LMB off
  set RMB on
end sub

sub endAllMotors
  set LMF off
  set LMB off
  set RMF off
  set RMB off
end sub

sub turnFanAndMotorsOn
  portb = 26
end sub

sub turnFanOn
  portb = 16
end sub

sub readPrintVal
  loadSensorVal

  'printing sensor values
  locate 0, 8
```

```
  print "R"
  print WDLVal
  locate 0, 12
  print "F"
  print WDFVal
  locate 1, 0
  print "FD"
  print FDVal
  locate 1, 8
  print "LD"
  print LDVal
  locate 1, 13
  print "C"
  print LDCount

end sub

sub wallHugging
  'wallhugging
      'if we see a wall in front of us in ANY circumstance
      if WDFVal < dist then
      pivotR
      'pause 100

      'if there is no wall in front of us but there is a wall
beside us
      else if WDLVal < dist then

        'adjusting to stay close to the wall
        'if we are too close to the wall, turn right
        if WDLVal < 10 then
        turnR

        else if WDLVal < 15 then
        moveF

        'if we are too far from the wall, turn left
        else
        turnL

        end if

      'if there is neither a wall in front nor beside us
      else
```

```
      turnL

      end if
end sub

  Do
  CLS
  locate 0, 0
  print "main"
  readPrintVal

  wallHugging

    'counting line detection
    'if the sensor does not detect a line
    if LDVal = 1 then
      lineAlreadyDetected = 0

    'if the sensor detects a line AND we have not already
counted this line
    else if LDVal = 0 and lineAlreadyDetected = 0 then
      lineAlreadyDetected = 1
      LDCount = LDCount + 1
      locate 1, 13
      print "C"
      print LDCount

    'if neither of these if statements trigger, we have already
counted the line we are over
    'therefore, do not do anything
    end if


    'if flame is detected, run fireFightingMode
    if flameDetected = 1 Then
      fireFightingMode
    end if

    'engage room 4 algorithm
    if LDCount >= 6 then
    AlgoRoomFour
    end if

  pause 100
```

```
  Loop

sub fireFightingMode
  Do
  'while flameDetected = 1
  CLS
  locate 0,0
  print "flame"
  readPrintVal

    if WDFVal > 20 then
      moveF
      pause 100

    else
      locate 0, 6
      print "fanOn"
      turnFanOn
      Do

        for c1 = 0 to 5
          pivotL
          turnFanOn
          pause 100
        next

        for c1 = 0 to 5
          pivotR
          turnFanOn
          pause 100
        next

      Loop
    end if


  Loop

end sub

sub AlgoRoomFour
  'hardcoded values to get into the fourth room
```

```
  'implement wallhugging for a bit longer tco get past the turn
  for c2 = 0 to 5
    readPrintVal
    wallHugging
    pause 100
  next

  turnL
  pause 2000
  moveF
  pause 2500
  turnR
  pause 600

  Do
  CLS
  'wallhug until we reach the flame
  readPrintVal
  locate 0, 0
  print "wallHug"
  wallHugging

      'if flame is detected, run fireFightingMode
    if flameDetected = 1 Then
      fireFightingMode
    end if


  Loop


end sub
```

Code for Maze 2 [INCOMPLETE]:

```
 'CHIP SETUP
#chip 16f887, 4
#config OSC = INTOSCIO
```

```
'LCD SETUP
#define LCD_IO_ 4
#define LCD_RS  PORTD.0
#define LCD_RW  PORTD.1
#define LCD_Enable PORTD.2
#define LCD_DB4 PORTD.4
#define LCD_DB5 PORTD.5
#define LCD_DB6 PORTD.6
#define LCD_DB7 PORTD.7
dir PORTD out

'MOTORS AND FAN SETUP
#define RMB PORTB.0
#define RMF PORTB.1
#define LMB PORTB.2
#define LMF PORTB.3
#define FAN PORTB.4
dir PORTB out

'SENSORS SETUP
#define WDF an5
#define WDR an6
#define FD an1
#define LD an0
dir PORTA in
dir PORTE in

'VARIABLES SETUP
dim FDVal as Byte
dim WDFVal as Byte
dim WDLVal as Byte
dim LDVal as Bit
dim LDCount as Byte
LDCount = 0
dim lineAlreadyDetected as Bit
lineAlreadyDetected = 0
dim dist as Byte
dist = 22
dim lastMove as String

  'returns 0 if the flame is not detected, 1 if the flame is
detected. flame threshold is 8
function flameDetected
    if FDVal > 8 then
```

```
      flameDetected = 0
    else
      flameDetected = 1
    end if
end Function

  'reads values from sensors and loads them into variables
sub loadSensorVal
  'SENSORS INPUT
  FDVal = ReadAD(FD)
  WDLVal = (((6787/(ReadAD(WDR)-3)))-4)/5
  WDFVal = (((6787/(ReadAD(WDF)-3)))-4)/5

  if ReadAD(LD) > 50 Then
    LDVal = 1
  else
    LDVal = 0
  end if

end sub

  'moves forward
sub moveF
  set LMF on
  set RMF on
  set LMB off
  set RMB off
  pause 100
end sub

  'moves backward
sub moveB
  set LMF off
  set RMF off
  set LMB on
  set RMB on
  pause 100
end sub

'note: PIVOT subfunctions are used for hard turns, while TURN
subfunctions are used for adjusting to stay close to the wall
  'pivots left [USED FOR TURNS]
sub pivotL
  set LMF off
```

```
  set RMF on
  set LMB on
  set RMB off
  pause 1000
end sub

  'pivots right [USED FOR TURNS]
sub pivotR
  set LMF on
  set RMF off
  set LMB off
  set RMB on
  pause 1000
end sub

  'turns left [USED FOR WALLHUGGING]
sub turnL
  set LMF off
  set RMF on
  set LMB off
  set RMB off
  pause 100
end sub

  'turns right [USED FOR WALLHUGGING]
sub turnR
  set LMF on
  set RMF off
  set LMB off
  set RMB off
  pause 100
end sub

sub endAllMotors
  set LMF off
  set LMB off
  set RMF off
  set RMB off
end sub

sub turnFanAndMotorsOn
  portb = 26
end sub
```

```
sub turnFanOn
  portb = 16
end sub

sub readPrintVal
  loadSensorVal

  locate 0, 0
  print lastMove
  'printing sensor values
  locate 0, 8
  print "R"
  print WDLVal
  locate 0, 12
  print "F"
  print WDFVal
  locate 1, 0
  print "FD"
  print FDVal
  locate 1, 8
  print "LD"
  print LDVal
  locate 1, 13
  print "C"
  print LDCount

end sub

sub wallHugging
  'wallhugging
      'if we see a wall in front of us in ANY circumstance
      if WDFVal < dist then
            lastMove = "RIGHT"
      pivotR

      'if there is no wall in front of us but there is a wall
beside us
      else if WDLVal < dist then

        'adjusting to stay close to the wall
        'if we are too close to the wall, turn right
        if WDLVal < 10 then
        turnR
```

```
        else if WDLVal < 15 then
        moveF

        'if we are too far from the wall, turn left
        else
        turnL

        end if
        lastMove = "FWRD"

      'if there is neither a wall in front nor beside us
      else
        repeat 5
          moveF
        end repeat
        pivotL
        lastMove = "LEFT"

      end if

end sub

sub fireFightingMode
  Do
  'while flameDetected = 1
  CLS
  locate 0,0
  print "flame"
  readPrintVal

    if WDFVal > 20 then
      moveF
      pause 100

    else
      turnL
      pause 700

      locate 0, 6
      print "fanOn"
      turnFanOn

      Do
        turnFanOn
```

```
        pivotL
        endAllMotors
        turnFanOn
        pause 500

      Loop
    end if


  Loop

end sub

'turns right and scans for flame. if flame is not detected,
then return to original positios
function scanForFlame

  Repeat 10
    CLS
    readPrintVal
    locate 0,0
    print "FSCAN"

    if FlameDetected = 1 Then
      scanForFlame = 1
      exit Function
    end if

    pivotR
    pause 100

  End Repeat

  repeat 10
    pivotL
  end repeat
scanForFlame = 0

end function

'code for up to room 1
sub room1

  'wallhugging until we have detected two lines
```

```
  Do while LDCount < 2
  CLS
  readPrintVal

      'counting line detection
    'if the sensor does not detect a line
    if LDVal = 1 then
      lineAlreadyDetected = 0

    'if the sensor detects a line AND we have not already
counted this line
    else if LDVal = 0 and lineAlreadyDetected = 0 then
      lineAlreadyDetected = 1
      LDCount = LDCount + 1
      locate 1, 13
      print "C"
      print LDCount

    'if neither of these if statements trigger, we have already
counted the line we are over
    'therefore, do not do anything
    end if

  wallhugging
  pause 100

  Loop

'back up once two line counts have been detected
  repeat 3
  moveB
  end repeat

  'turn and scan for flame
  'if we see the flame, move forward and put it out
  if scanForFlame = 1 then

  CLS
  readPrintVal

    repeat 10
    moveF
    end repeat
```

```
      Do while WDFVal < dist
      CLS
      readPrintVal

      wallhugging

        if flameDetected = 1 then
          fireFightingMode
        end if
      pause 100

      Loop

  end if

end sub

sub room2

  Do while

  Loop
end sub
```