```
/************************************************
  Mitchell Williams

  The Rat Pack - Rat Track Version 1.01 Updated 4/24/2019

  This code will allow a 7" LCD TFT Touchscreen to interface with a stepper motor.
  The speed of the stepper motor, and the amount of time to run, is controllable.
  Developed for joint research between Idaho State University and University of Idaho
  Biological Engineering departments.

  ---Additional Libraries and Credit----------
  UTFT and UTouch Libraries and Examples - Henning Karlsen, http://rinkydinkelectronics.com/
  dtostrf Workaround for Arduino DUE -     Cristian Maglie <c.maglie@bug.st>
************************************************/

#include <UTFT.h>
#include <UTouch.h>

// ----------------------------------------------------------
// Arduino DUE Shield setup
UTFT myGLCD(SSD1963_800480, 38, 39, 40, 41); //(byte model, int RS, int WR, int CS, int RST, int SER)
UTouch  myTouch( 43, 42, 44, 45, 46);  //byte tclk, byte tcs, byte din, byte dout, byte irq

//Pin Setup
int pulPin = 7; // Pulse for Stepper Driver
int dirPin = 6; // Direction for Stepper Driver
int optoPin = 5; // Opto-Coupler Pin for Stepper Driver
int enblPin = 4; // Enable Pin for Stepper Driver
int fangndPin = 3; // Fan GND
```

```
int fanPin = 2; // Fan Power Pin

//--------------------------------------


//Stepper Motor Timing

unsigned long one_second = 1000000; // One second in microseconds

float pulses_per_rev = 800; // Pulses per revolution, set using SW1 - SW3 on DM320T Stepper Driver

float speedval = 7.5; // Linear velocity, cm/s   (min:0.5 max:15)

unsigned long stepper_delay = (8808.6 / pow(speedval, 1.017)); // Delay between steps in microseconds. Calibrated using using slow-motion film as reference

//unsigned long stepper_delay = (6982189.673 / (speedval*pulses_per_rev)); // Delay between steps in microseconds. Calculated radius of shaft = 1.11125cm

unsigned long test_stepper_delay = 106162.9615; // Test delay

//--------------------------------------


//Timer

int x, y;

int timerposition = 0;

long timerNumeric[5] = {0, 0, 0, 0, 0};

String timerStr[5] = {String('-'), String('-'), String('-'), String('-'), String('-')};

String prevtimerStr[5] = {String('-'), String('-'), String('-'), String('-'), String('-')};

long timeInSeconds = 0;

int timeHour = 0;

int timeTenMin = 0;

int timeMin = 0;

int timeTenSec = 0;

int timeSec = 0;

//--------------------------------------


// Status Check
```

```
String statusCheck = String("canceled");
// "canceled" mode refers to PAUSED, but control ENABLED
// "paused" mode refers to PAUSED,  control DISABLED
// "running" mode refers to RUNNING, control DISABLED
//--------------------------------------

// Screen and Button Variables

// Declaring fonts that will be used in the RATPACK UI
extern uint8_t GroteskBold16x32[];
extern uint8_t GroteskBold24x48[];
extern uint8_t GroteskBold32x64[];
extern uint8_t TinyFont[];

float stdButtonSize[2] = {70, 60};
float stdButtonFillColor[3] = {0, 204, 255};
float voidButtonFillColor[3] = {190, 190, 190};
float dispButtonFillColor[3] = {98, 157, 172};
float stdButtonFrameColor[3] = {255, 255, 255};

float goButtonFillColor[3] = {0, 204, 0};
float pauseButtonFillColor[3] = {255, 219, 77};
float cancelButtonFillColor[3] = {204, 0, 0};

//(screen size is (X:800, Y:480)

// Timer Size & Positions
float timeButtonSize = 50;
float timenumGap = 10;
```

```
float timeEnterButtonSize[2] = {140, 70};

float timeDispButtonSize[2] = {200, 60};

float timePos[2] = {600, 100};

float timeNumSize[2] = {168 + 30, 48 + 20};

float timeNumPos[2] = {timePos[0] + 1.5 * timeButtonSize + timenumGap, timePos[1] + 4 *
timeButtonSize + 4 * timenumGap + .5 * timeNumSize[1]};


// Speed Button Size & Positions

float speedButtonSize[2] = {200, 60};


float speedButtonPos[2] = {(180 + 0), 150};

float speedDOWNButtonPos[2] = {(speedButtonPos[0] - 40), speedButtonPos[1] + 70};

float speedDOWNDOWNButtonPos[2] = {(speedButtonPos[0] - 120), speedButtonPos[1] + 70};

float speedUPButtonPos[2] = {(speedButtonPos[0] + 40), speedButtonPos[1] + 70};

float speedUPUPButtonPos[2] = {(speedButtonPos[0] + 120), speedButtonPos[1] + 70};


// Go/Pause/Cancel Button Size & Positions

float goButtonSize[2] = {100, 100};

float pauseButtonSize[2] = {100, 100};

float cancelButtonSize[2] = {100, 100};


float goButtonPos[2] = {(340), 410};

float pauseButtonPos[2] = {(goButtonPos[0] - 120), goButtonPos[1]};

float cancelButtonPos[2] = {(goButtonPos[0] + 120), goButtonPos[1]};


float fontWidth = 24;

float fontHeight = 48;


char *dtostrf (double val, signed char width, unsigned char prec, char *sout) {
```

```
  char fmt[20];
  sprintf(fmt, "%%%d.%df", width, prec);
  sprintf(sout, fmt, val);
  return sout;
}
//------------------------------------------------------------------------------

void drawButtons(float color1, float color2, float color3)
{
  myGLCD.setFont(GroteskBold24x48);
  float fontWidth = myGLCD.getFontXsize();
  float fontHeight = myGLCD.getFontYsize();

  // Fill SPEED button rectangles
  myGLCD.setColor(color1, color2, color3);
  myGLCD.fillRoundRect(speedDOWNButtonPos[0] - .5 * stdButtonSize[0] , speedDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNButtonPos[0] + .5 * stdButtonSize[0] , speedDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <
  myGLCD.fillRoundRect(speedDOWNDOWNButtonPos[0] - .5 * stdButtonSize[0] , speedDOWNDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNDOWNButtonPos[0] + .5 * stdButtonSize[0] , speedDOWNDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <<
  myGLCD.fillRoundRect(speedUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPButtonPos[1] - .5 * stdButtonSize[1], speedUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPButtonPos[1] + .5 * stdButtonSize[1]); // >
  myGLCD.fillRoundRect(speedUPUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPUPButtonPos[1] - .5 * stdButtonSize[1], speedUPUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPUPButtonPos[1] + .5 * stdButtonSize[1]); // >>

  // Draw TIMER Buttons
  for (x = 0; x < 3; x++) { //UPPER ROW #7 #8 #9
    myGLCD.setColor(color1, color2, color3);
```

```
  myGLCD.fillRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1], timePos[0] +
timeButtonSize + x * timeButtonSize + x * timenumGap, timePos[1] + timeButtonSize);

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1], timePos[0] +
timeButtonSize + x * timeButtonSize + x * timenumGap, timePos[1] + timeButtonSize);

 }

 for (x = 0; x < 3; x++) { //MIDDLE ROW #4 #5 #6

  myGLCD.setColor(color1, color2, color3);

  myGLCD.fillRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1] + timeButtonSize
+ timenumGap, timePos[0] + timeButtonSize + x * (timeButtonSize + timenumGap), timePos[1] + 2 *
timeButtonSize + timenumGap);

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1] +
timeButtonSize + timenumGap, timePos[0] + timeButtonSize + x * (timeButtonSize + timenumGap),
timePos[1] + 2 * timeButtonSize + timenumGap);

 }

 for (x = 0; x < 3; x++) { //LOWER ROW #1 #2 #3

  myGLCD.setColor(color1, color2, color3);

  myGLCD.fillRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1] + 2 *
timeButtonSize + 2 * timenumGap, timePos[0] + timeButtonSize + x * (timeButtonSize + timenumGap),
timePos[1] + 3 * timeButtonSize + 2 * timenumGap);

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(timePos[0] + x * (timeButtonSize + timenumGap), timePos[1] + 2 *
timeButtonSize + 2 * timenumGap, timePos[0] + timeButtonSize + x * (timeButtonSize + timenumGap),
timePos[1] + 3 * timeButtonSize + 2 * timenumGap);

  myGLCD.setColor(color1, color2, color3);

 }

 // #0

 myGLCD.setColor(color1, color2, color3); // #0

 myGLCD.fillRoundRect(timePos[0], timePos[1] + 3 * timeButtonSize + 3 * timenumGap, timePos[0] +
timeButtonSize, timePos[1] + 4 * timeButtonSize + 3 * timenumGap);

 myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);
```

```
  myGLCD.drawRoundRect(timePos[0], timePos[1] + 3 * timeButtonSize + 3 * timenumGap, timePos[0] + timeButtonSize, timePos[1] + 4 * timeButtonSize + 3 * timenumGap);


  //FRAME

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(speedDOWNButtonPos[0] - .5 * stdButtonSize[0] , speedDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNButtonPos[0] + .5 * stdButtonSize[0] , speedDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <

  myGLCD.drawRoundRect(speedDOWNDOWNButtonPos[0] - .5 * stdButtonSize[0] , speedDOWNDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNDOWNButtonPos[0] + .5 * stdButtonSize[0] , speedDOWNDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <<

  myGLCD.drawRoundRect(speedUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPButtonPos[1] - .5 * stdButtonSize[1], speedUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPButtonPos[1] + .5 * stdButtonSize[1]); // >

  myGLCD.drawRoundRect(speedUPUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPUPButtonPos[1] - .5 * stdButtonSize[1], speedUPUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPUPButtonPos[1] + .5 * stdButtonSize[1]); // >>


  // Add button TEXT

  myGLCD.setColor(255, 255, 255);

  myGLCD.setBackColor(color1, color2, color3);  // SPEED

  myGLCD.print("<", speedDOWNButtonPos[0] - 1 * .5 * fontWidth, speedDOWNButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.print("<<", speedDOWNDOWNButtonPos[0] - 2 * .5 * fontWidth, speedDOWNDOWNButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.print(">", speedUPButtonPos[0] - 1 * .5 * fontWidth, speedUPButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.print(">>", speedUPUPButtonPos[0] - 2 * .5 * fontWidth, speedUPUPButtonPos[1] - 1 * .5 * fontHeight);


  myGLCD.setBackColor(color1, color2, color3);

  myGLCD.print("7", timePos[0] + .5 * timeButtonSize - 1 * .5 * fontWidth, timePos[1] + .5 * timeButtonSize - 1 * .5 * fontHeight);     //#7
```

```
  myGLCD.print("8", timePos[0] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontWidth, timePos[1] + .5 * timeButtonSize - 1 * .5 * fontHeight); //#8

  myGLCD.print("9", timePos[0] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontWidth, timePos[1] + .5 * timeButtonSize - 1 * .5 * fontHeight); //#9

  myGLCD.print("4", timePos[0] + .5 * timeButtonSize - 1 * .5 * fontWidth, timePos[1] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontHeight);     //#4

  myGLCD.print("5", timePos[0] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontWidth, timePos[1] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontHeight); //#5

  myGLCD.print("6", timePos[0] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontWidth, timePos[1] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontHeight); //#6

  myGLCD.print("1", timePos[0] + .5 * timeButtonSize - 1 * .5 * fontWidth, timePos[1] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontHeight);   //#1

  myGLCD.print("2", timePos[0] + 1.5 * timeButtonSize + timenumGap - 1 * .5 * fontWidth, timePos[1] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontHeight); //#2

  myGLCD.print("3", timePos[0] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontWidth, timePos[1] + 2.5 * timeButtonSize + 2 * timenumGap - 1 * .5 * fontHeight); //#3

  myGLCD.print("0", timePos[0] + .5 * timeButtonSize - 1 * .5 * fontWidth, timePos[1] + 3.5 * timeButtonSize + 3 * timenumGap - 1 * .5 * fontHeight); //#0

}


void drawStartup() // Draw Startup Screen

{

  //UNIVERSITY LOGO

  int universityItop[2] = {100, 60};

  int universityImid[2] = {64, 100};

  int universityIgap = 8;


  int universityImidPos[2] = {400, 130};

  int universityItopPos[2] = {400, universityImidPos[1] - .5 * universityImid[1] - universityItop[1]};

  int universityIbotPos[2] = {400, universityImidPos[1] + .5 * universityImid[1] + universityItop[1]};


  myGLCD.setColor(252, 186, 4);
```

```
  myGLCD.fillRect(universityItopPos[0] - .5 * universityItop[0], universityItopPos[1], universityItopPos[0] + .5 * universityItop[0], universityItopPos[1] + universityItop[1]);

  myGLCD.fillRect(universityIbotPos[0] - .5 * universityItop[0], universityIbotPos[1], universityIbotPos[0] + .5 * universityItop[0], universityIbotPos[1] - universityItop[1]);

  myGLCD.fillRect(universityImidPos[0] - .5 * universityImid[0], universityImidPos[1] - .5 * universityImid[1], universityImidPos[0] + .5 * universityImid[0], universityImidPos[1] + .5 * universityImid[1]);

  myGLCD.setColor(0, 0, 0);

  myGLCD.fillRect(universityItopPos[0] - .5 * universityItop[0] + universityIgap, universityItopPos[1] + universityIgap, universityItopPos[0] + .5 * universityItop[0] - universityIgap, universityItopPos[1] + universityItop[1] - universityIgap);

  myGLCD.fillRect(universityIbotPos[0] - .5 * universityItop[0] + universityIgap, universityIbotPos[1] - universityIgap, universityIbotPos[0] + .5 * universityItop[0] - universityIgap, universityIbotPos[1] - universityItop[1] + universityIgap);

  myGLCD.fillRect(universityImidPos[0] - .5 * universityImid[0] + universityIgap, universityImidPos[1] - .5 * universityImid[1] - universityIgap, universityImidPos[0] + .5 * universityImid[0] - universityIgap, universityImidPos[1] + .5 * universityImid[1] + universityIgap);

  myGLCD.setColor(252, 186, 4);

  myGLCD.fillRect(universityItopPos[0] - .5 * universityItop[0] + 2 * universityIgap, universityItopPos[1] + 2 * universityIgap, universityItopPos[0] + .5 * universityItop[0] - 2 * universityIgap, universityItopPos[1] + universityItop[1] - 2 * universityIgap);

  myGLCD.fillRect(universityIbotPos[0] - .5 * universityItop[0] + 2 * universityIgap, universityIbotPos[1] - 2 * universityIgap, universityIbotPos[0] + .5 * universityItop[0] - 2 * universityIgap, universityIbotPos[1] - universityItop[1] + 2 * universityIgap);

  myGLCD.fillRect(universityImidPos[0] - .5 * universityImid[0] + 2 * universityIgap, universityImidPos[1] - .5 * universityImid[1] - 2 * universityIgap, universityImidPos[0] + .5 * universityImid[0] - 2 * universityIgap, universityImidPos[1] + .5 * universityImid[1] + 2 * universityIgap);


  myGLCD.setColor(255, 255, 255);

  myGLCD.setBackColor(0, 0, 0);

  myGLCD.setFont(GroteskBold32x64);

  myGLCD.print("Rat Treadmill Control UI", CENTER, 250);

  myGLCD.setFont(GroteskBold16x32);

  delay(250);
```

```
  myGLCD.print("Version 1.01", CENTER, 330);
  myGLCD.print("University of Idaho", CENTER, 370);
  delay(3000);
  myGLCD.setColor(0, 0, 0);
  myGLCD.fillRect(0, 0, 800, 480);
}

void setup()
{
  pinMode(8, OUTPUT);  // Screen Backlight
  pinMode(pulPin, OUTPUT); // Stepper Pulse Pin
  pinMode(dirPin, OUTPUT); // Stepper Direction Pin
  pinMode(optoPin, OUTPUT); // Stepper Opto-Coupler Pin
  pinMode(enblPin, OUTPUT); // Stepper Enable Pin
  pinMode(fangndPin, OUTPUT); // Fan GND
  pinMode(fanPin, OUTPUT); // Fan Power

  digitalWrite(8, HIGH);// Turns screen Backlight ON
  digitalWrite(pulPin, LOW); // Sets Pulse LOW by default
  digitalWrite(optoPin, HIGH); // Opto-Coupler pin set HIGH to turn on
  digitalWrite(enblPin, LOW); // Enable LOW to initially disable stepper motor
  digitalWrite(dirPin, LOW); // Direction Pin (LOW = CW Motion)
  digitalWrite(fangndPin, LOW); // GND for Fan set LOW
  digitalWrite(fanPin, LOW); // Fan initially LOW, disabled
  // ------------------------------------------------------------

  // Initial setup
  myGLCD.InitLCD();
  myGLCD.clrScr();
```

```
  myTouch.InitTouch();
  myTouch.setPrecision(PREC_HI);

  // Draw STARTUP
  drawStartup();

  // Draw TITLE
  myGLCD.setFont(GroteskBold32x64);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("Rat Track", CENTER, 20);
  drawButtons(stdButtonFillColor[0], stdButtonFillColor[1], stdButtonFillColor[2]);

  myGLCD.setFont(GroteskBold24x48);
  float fontWidth = myGLCD.getFontXsize();
  float fontHeight = myGLCD.getFontYsize();

  // SPEED DISPLAY
  myGLCD.setColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]); // TIMEDISPLAY
  myGLCD.fillRoundRect(speedButtonPos[0] - .5 * speedButtonSize[0] , speedButtonPos[1] - .5 * speedButtonSize[1], speedButtonPos[0] + .5 * speedButtonSize[0] , speedButtonPos[1] + .5 * speedButtonSize[1]); // SPEED#

  // TIME DISPLAY
  myGLCD.setColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]); // TIMEDISPLAY
  myGLCD.fillRoundRect(timeNumPos[0] - .5 * timeNumSize[0], timeNumPos[1] - .5 * timeNumSize[1], timeNumPos[0] + .5 * timeNumSize[0], timeNumPos[1] + .5 * timeNumSize[1]);
  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);
  myGLCD.drawRoundRect(timeNumPos[0] - .5 * timeNumSize[0], timeNumPos[1] - .5 * timeNumSize[1], timeNumPos[0] + .5 * timeNumSize[0], timeNumPos[1] + .5 * timeNumSize[1]);
```

```
  updateTimeDisp(255, 255, 255);


  // DEL

  myGLCD.setColor(cancelButtonFillColor[0], cancelButtonFillColor[1], cancelButtonFillColor[2]); // DEL

  myGLCD.fillRoundRect(timePos[0] + timeButtonSize + timenumGap, timePos[1] + 3 * timeButtonSize +
3 * timenumGap, timePos[0] + 3 * timeButtonSize + 2 * timenumGap, timePos[1] + 4 * timeButtonSize +
3 * timenumGap);

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(timePos[0] + timeButtonSize + timenumGap, timePos[1] + 3 * timeButtonSize
+ 3 * timenumGap, timePos[0] + 3 * timeButtonSize + 2 * timenumGap, timePos[1] + 4 * timeButtonSize
+ 3 * timenumGap);


  // Draw GO, PAUSE, CANCEL buttons

  myGLCD.setColor(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]);

  myGLCD.fillRoundRect(goButtonPos[0] - .5 * goButtonSize[0] , goButtonPos[1] - .5 * goButtonSize[1],
goButtonPos[0] + .5 * goButtonSize[0] , goButtonPos[1] + .5 * goButtonSize[1]); // GO

  myGLCD.setColor(pauseButtonFillColor[0], pauseButtonFillColor[1], pauseButtonFillColor[2]);

  myGLCD.fillRoundRect(pauseButtonPos[0] - .5 * pauseButtonSize[0] , pauseButtonPos[1] - .5 *
pauseButtonSize[1], pauseButtonPos[0] + .5 * pauseButtonSize[0] , pauseButtonPos[1] + .5 *
pauseButtonSize[1]); // PAUSE

  myGLCD.setColor(cancelButtonFillColor[0], cancelButtonFillColor[1], cancelButtonFillColor[2]);

  myGLCD.fillRoundRect(cancelButtonPos[0] - .5 * cancelButtonSize[0] , cancelButtonPos[1] - .5 *
cancelButtonSize[1], cancelButtonPos[0] + .5 * cancelButtonSize[0] , cancelButtonPos[1] + .5 *
cancelButtonSize[1]); // CANCEL


  // Draw Button FRAMES

  myGLCD.setColor(stdButtonFrameColor[0], stdButtonFrameColor[1], stdButtonFrameColor[2]);

  myGLCD.drawRoundRect(speedButtonPos[0] - .5 * speedButtonSize[0] , speedButtonPos[1] - .5 *
speedButtonSize[1], speedButtonPos[0] + .5 * speedButtonSize[0] , speedButtonPos[1] + .5 *
speedButtonSize[1]); // SPEED#

  myGLCD.drawRoundRect(goButtonPos[0] - .5 * goButtonSize[0] , goButtonPos[1] - .5 * goButtonSize[1],
goButtonPos[0] + .5 * goButtonSize[0] , goButtonPos[1] + .5 * goButtonSize[1]); // GO
```

```
  myGLCD.drawRoundRect(pauseButtonPos[0] - .5 * pauseButtonSize[0] , pauseButtonPos[1] - .5 * pauseButtonSize[1], pauseButtonPos[0] + .5 * pauseButtonSize[0] , pauseButtonPos[1] + .5 * pauseButtonSize[1]); // PAUSE

  myGLCD.drawRoundRect(cancelButtonPos[0] - .5 * cancelButtonSize[0] , cancelButtonPos[1] - .5 * cancelButtonSize[1], cancelButtonPos[0] + .5 * cancelButtonSize[0] , cancelButtonPos[1] + .5 * cancelButtonSize[1]); // CANCEL


  // Add Button Text

  myGLCD.setColor(255, 255, 255);

  myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);

  myGLCD.print("7.5", speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.print("cm/s", speedButtonPos[0] + 0 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 * fontHeight);


  myGLCD.setBackColor(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]);  // GO/PAUSE/CANCEL

  myGLCD.print("GO", goButtonPos[0] - 2 * .5 * fontWidth, goButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.setBackColor(pauseButtonFillColor[0], pauseButtonFillColor[1], pauseButtonFillColor[2]);

  myGLCD.print("||", pauseButtonPos[0] - 2 * .5 * fontWidth, pauseButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.setBackColor(cancelButtonFillColor[0], cancelButtonFillColor[1], cancelButtonFillColor[2]);

  myGLCD.print("X", cancelButtonPos[0] - 1 * .5 * fontWidth, cancelButtonPos[1] - 1 * .5 * fontHeight);

  myGLCD.print("DEL", timePos[0] + 2 * timeButtonSize + 1.5 * timenumGap - 3 * .5 * fontWidth, timePos[1] + 3.5 * timeButtonSize + 3 * timenumGap - 1 * .5 * fontHeight); //TIME DEL


  myGLCD.setBackColor (0, 0, 0);  // Reset background back to black

  myGLCD.setFont(GroteskBold16x32);

  myGLCD.print("Status:", goButtonPos[0] - 7 * 24, goButtonPos[1] - 90);

  myGLCD.setColor(255, 255, 255); //TEXT

  myGLCD.print("WAITING", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);

  myGLCD.setFont(GroteskBold24x48);

}
```

```
void loop()

{

  while (statusCheck != "running")

  {

    if (myTouch.dataAvailable())

    {

      myTouch.read();

      x = myTouch.getX();

      y = myTouch.getY();

      //-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

      if (((x >= speedDOWNButtonPos[0] - .5 * stdButtonSize[0]) && (x <= speedDOWNButtonPos[0] + .5 * stdButtonSize[0])) && ((y >= speedDOWNButtonPos[1] - .5 * stdButtonSize[1]) && (y <= speedDOWNButtonPos[1] + .5 * stdButtonSize[1])) && (statusCheck != "paused") ) // <

      { //SPEED DOWN BUTTON

        drawFrame(speedDOWNButtonPos[0] - .5 * stdButtonSize[0] , speedDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNButtonPos[0] + .5 * stdButtonSize[0] , speedDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <

        speedval = speedval - 0.1;

        if (speedval <= 0.5)

        {

          speedval = 0.5;

        }

        else

        {

          speedval = speedval;

        }

        char speedvaldecimal[5] = {0, 0, 0, 0, 0};  // Converts float to a single decimal point

        dtostrf(speedval, 3, 1, speedvaldecimal);
```

```
      myGLCD.setColor(255, 255, 255); //TEXT

      myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);

      myGLCD.print("    ", speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight); // Cover old value

      myGLCD.print(speedvaldecimal, speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5
* fontHeight); // Print new value

      myGLCD.print("cm/s", speedButtonPos[0] + 0 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight);

    }

    //---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------

    if (((x >= speedDOWNDOWNButtonPos[0] - .5 * stdButtonSize[0]) && (x <=
speedDOWNDOWNButtonPos[0] + .5 * stdButtonSize[0])) && ((y >= speedDOWNDOWNButtonPos[1] -
.5 * stdButtonSize[1]) && (y <= speedDOWNDOWNButtonPos[1] + .5 * stdButtonSize[1])) &&
(statusCheck != "paused") && (statusCheck != "paused") ) // <<

    { //SPEED DOWNDOWN BUTTON

      drawFrame(speedDOWNDOWNButtonPos[0] - .5 * stdButtonSize[0] ,
speedDOWNDOWNButtonPos[1] - .5 * stdButtonSize[1], speedDOWNDOWNButtonPos[0] + .5 *
stdButtonSize[0] , speedDOWNDOWNButtonPos[1] + .5 * stdButtonSize[1]); // <<

      speedval = speedval - 1;

      if (speedval <= 0.5)

      {

        speedval = 0.5;

      }

      else

      {

        speedval = speedval;

      }


      char speedvaldecimal[5] = {0, 0, 0, 0, 0};  // Converts float to a single decimal point

      dtostrf(speedval, 3, 1, speedvaldecimal);
```

```
      myGLCD.setColor(255, 255, 255); //TEXT

      myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);

      myGLCD.print("    ", speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight); // Cover old value

      myGLCD.print(speedvaldecimal, speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5
* fontHeight); // Print new value

      myGLCD.print("cm/s", speedButtonPos[0] + 0 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight);

    }

    //--------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------

    if (((x >= speedUPButtonPos[0] - .5 * stdButtonSize[0]) && (x <= speedUPButtonPos[0] + .5 *
stdButtonSize[0])) && ((y >= speedUPButtonPos[1] - .5 * stdButtonSize[1]) && (y <=
speedUPButtonPos[1] + .5 * stdButtonSize[1])) && (statusCheck != "paused") ) // >

    { //SPEED UP BUTTON

      drawFrame(speedUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPButtonPos[1] - .5 *
stdButtonSize[1], speedUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPButtonPos[1] + .5 *
stdButtonSize[1]); // >

      speedval = speedval + 0.1;

      if (speedval >= 15)

      {

        speedval = 15;

      }

      else

      {

        speedval = speedval;

      }


      char speedvaldecimal[5] = {0, 0, 0, 0, 0};  // Converts float to a single decimal point

      dtostrf(speedval, 3, 1, speedvaldecimal);
```

```
      myGLCD.setColor(255, 255, 255); //TEXT

      myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);

      myGLCD.print("    ", speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight); // Cover old value

      myGLCD.print(speedvaldecimal, speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5
* fontHeight); // Print new value

      myGLCD.print("cm/s", speedButtonPos[0] + 0 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight);

    }

    //-----------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------

    if (((x >= speedUPUPButtonPos[0] - .5 * stdButtonSize[0]) && (x <= speedUPUPButtonPos[0] + .5 *
stdButtonSize[0])) && ((y >= speedUPUPButtonPos[1] - .5 * stdButtonSize[1]) && (y <=
speedUPUPButtonPos[1] + .5 * stdButtonSize[1])) && (statusCheck != "paused") ) // >>

    { //SPEED UPUP BUTTON

      drawFrame(speedUPUPButtonPos[0] - .5 * stdButtonSize[0] , speedUPUPButtonPos[1] - .5 *
stdButtonSize[1], speedUPUPButtonPos[0] + .5 * stdButtonSize[0] , speedUPUPButtonPos[1] + .5 *
stdButtonSize[1]); // >>

      speedval = speedval + 1;

      if (speedval >= 15)

      {

        speedval = 15;

      }

      else

      {

        speedval = speedval;

      }

      char speedvaldecimal[5] = {0, 0, 0, 0, 0};  // Converts float to a single decimal point

      dtostrf(speedval, 3, 1, speedvaldecimal);


      myGLCD.setColor(255, 255, 255); //TEXT

      myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);
```

```
      myGLCD.print("    ", speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight); // Cover old value

      myGLCD.print(speedvaldecimal, speedButtonPos[0] - 8 * .5 * fontWidth, speedButtonPos[1] - 1 * .5
* fontHeight); // Print new value

      myGLCD.print("cm/s", speedButtonPos[0] + 0 * .5 * fontWidth, speedButtonPos[1] - 1 * .5 *
fontHeight);

    }

    //-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------

    if (((x >= goButtonPos[0] - .5 * goButtonSize[0]) && (x <= goButtonPos[0] + .5 * goButtonSize[0])) &&
((y >= goButtonPos[1] - .5 * goButtonSize[1]) && (y <= goButtonPos[1] + .5 * goButtonSize[1]))) // GO

    { //GO BUTTON

      drawFrame(goButtonPos[0] - .5 * goButtonSize[0] , goButtonPos[1] - .5 * goButtonSize[1],
goButtonPos[0] + .5 * goButtonSize[0] , goButtonPos[1] + .5 * goButtonSize[1]); // GO

      myGLCD.setBackColor (0, 0, 0);

      myGLCD.setFont(GroteskBold16x32);

      myGLCD.print("       ", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);

      myGLCD.setColor(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]); //TEXT

      myGLCD.print("RUNNING", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);


      drawButtons(voidButtonFillColor[0], voidButtonFillColor[1], voidButtonFillColor[2]);

      if (timeInSeconds > 0)

      {

        timeInSeconds = timeInSeconds;

        timeHour = timeInSeconds / 3600;

        timeTenMin = (timeInSeconds % 3600) / 600;

        timeMin = (timeInSeconds % 3600) / 60 - timeTenMin * 10;

        timeTenSec = (timeInSeconds % 3600) % 60 / 10;

        timeSec = (timeInSeconds % 3600) % 60 - timeTenSec * 10;

        timerStr[0] = String(timeHour);   //HOUR

        timerStr[1] = String(timeTenMin); //TEN MINUTES
```

```
  timerStr[2] = String(timeMin);    //MINUTES
  timerStr[3] = String(timeTenSec); //TEN SECONDS
  timerStr[4] = String(timeSec);    //SECONDS
}
else
{
  timeInSeconds = 3600; // If no time entered, defaults to 1 hour
  timerStr[0] = String('1');
  timerStr[1] = String('0');
  timerStr[2] = String('0');
  timerStr[3] = String('0');
  timerStr[4] = String('0');
}

if (statusCheck != "paused")
{
  updateTimeDisp(255, 255, 255);
}

myGLCD.setFont(TinyFont);
myGLCD.setColor(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]);
myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);
myGLCD.print(timerStr[0], timeNumPos[0] - 3.5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(":",    timeNumPos[0] - 2.5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(timerStr[1], timeNumPos[0]  - 1.5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(timerStr[2], timeNumPos[0] - .5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(":",    timeNumPos[0] + .5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(timerStr[3], timeNumPos[0] + 1.5 * 8, timeNumPos[1] + 3 * 8);
myGLCD.print(timerStr[4], timeNumPos[0] + 2.5 * 8, timeNumPos[1] + 3 * 8);
```

```
      statusCheck = String("running");
      stepper_delay = (8808.6 / pow(speedval, 1.017));
      digitalWrite(fanPin, HIGH); // Enables DC-fan
      digitalWrite(enblPin, HIGH); // Enables stepper motor
      steps();
    }

    //--------------------------------------------------------------------------------------------------------------------------------------------
    if (((x >= cancelButtonPos[0] - .5 * cancelButtonSize[0]) && (x <= cancelButtonPos[0] + .5 * cancelButtonSize[0])) && ((y >= cancelButtonPos[1] - .5 * cancelButtonSize[1]) && (y <= cancelButtonPos[1] + .5 * cancelButtonSize[1])) && (statusCheck != "canceled")) // CANCEL
    { //CANCEL BUTTON
      drawFrame(cancelButtonPos[0] - .5 * cancelButtonSize[0] , cancelButtonPos[1] - .5 * cancelButtonSize[1], cancelButtonPos[0] + .5 * cancelButtonSize[0] , cancelButtonPos[1] + .5 * cancelButtonSize[1]); // CANCEL
      cancel();
    }
    //--------------------------------------------------------------------------------------------------------------------------------------------
    if ((y >= (timePos[1])) && (y <= (timePos[1] + timeButtonSize)) && (statusCheck != "paused") ) // UPPER ROW
    { // UPPER ROW
      if ((x >= (timePos[0])) && (x <= (timePos[0] + timeButtonSize))) // #7
      {
        drawFrame(timePos[0], timePos[1], timePos[0] + timeButtonSize, timePos[1] + timeButtonSize);
        updateStr(7);
      }
      if ((x >= timePos[0] + timeButtonSize + timenumGap) && (x <= timePos[0] + 2 * timeButtonSize + timenumGap)) // #8
      {
```

```
        drawFrame(timePos[0] + timeButtonSize + timenumGap, timePos[1], timePos[0] + 2 *
timeButtonSize + timenumGap, timePos[1] + timeButtonSize);

        updateStr(8);

      }

      if ((x >= timePos[0] + 2 * timeButtonSize + 2 * timenumGap) && (x <= timePos[0] + 3 *
timeButtonSize + 2 * timenumGap)) // #9

      {

        drawFrame(timePos[0] + 2 * timeButtonSize + 2 * timenumGap, timePos[1], timePos[0] + 3 *
timeButtonSize + 2 * timenumGap, timePos[1] + timeButtonSize);

        updateStr(9);

      }

    }

    //-----------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------

    if ((y >= (timePos[1] + timeButtonSize + timenumGap)) && (y <= (timePos[1] + 2 * timeButtonSize +
timenumGap)) && (statusCheck != "paused") ) // MIDDLE ROW

    { // MIDDLE ROW

      if ((x >= (timePos[0])) && (x <= (timePos[0] + timeButtonSize))) // #4

      {

        drawFrame(timePos[0], timePos[1] + timeButtonSize + timenumGap, timePos[0] + timeButtonSize,
timePos[1] + 2 * timeButtonSize + timenumGap);

        updateStr(4);

      }

      if ((x >= timePos[0] + timeButtonSize + timenumGap) && (x <= timePos[0] + 2 * timeButtonSize +
timenumGap)) // #5

      {

        drawFrame(timePos[0] + timeButtonSize + timenumGap, timePos[1] + timeButtonSize +
timenumGap, timePos[0] + 2 * timeButtonSize + timenumGap, timePos[1] + 2 * timeButtonSize +
timenumGap);

        updateStr(5);

      }
```

```
      if ((x >= timePos[0] + 2 * timeButtonSize + 2 * timenumGap) && (x <= timePos[0] + 3 *
timeButtonSize + 2 * timenumGap)) // #6

      {

        drawFrame(timePos[0] + 2 * timeButtonSize + 2 * timenumGap, timePos[1] + timeButtonSize +
timenumGap, timePos[0] + 3 * timeButtonSize + 2 * timenumGap, timePos[1] + 2 * timeButtonSize +
timenumGap);

        updateStr(6);

      }

    }

    //-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------

    if ((y >= (timePos[1] + 2 * timeButtonSize + 2 * timenumGap)) && (y <= (timePos[1] + 3 *
timeButtonSize + 2 * timenumGap)) && (statusCheck != "paused") ) // LOWER ROW

    { // LOWER ROW

      if ((x >= (timePos[0])) && (x <= (timePos[0] + timeButtonSize))) // #1

      {

        drawFrame(timePos[0], timePos[1] + 2 * timeButtonSize + 2 * timenumGap, timePos[0] +
timeButtonSize, timePos[1] + 3 * timeButtonSize + 2 * timenumGap);

        updateStr(1);

      }

      if ((x >= timePos[0] + timeButtonSize + timenumGap) && (x <= timePos[0] + 2 * timeButtonSize +
timenumGap)) // #2

      {

        drawFrame(timePos[0] + timeButtonSize + timenumGap, timePos[1] + 2 * timeButtonSize + 2 *
timenumGap, timePos[0] + 2 * timeButtonSize + timenumGap, timePos[1] + 3 * timeButtonSize + 2 *
timenumGap);

        updateStr(2);

      }

      if ((x >= timePos[0] + 2 * timeButtonSize + 2 * timenumGap) && (x <= timePos[0] + 3 *
timeButtonSize + 2 * timenumGap)) // #3

      {
```

```
        drawFrame(timePos[0] + 2 * timeButtonSize + 2 * timenumGap, timePos[1] + 2 * timeButtonSize +
2 * timenumGap, timePos[0] + 3 * timeButtonSize + 2 * timenumGap, timePos[1] + 3 * timeButtonSize +
2 * timenumGap);

        updateStr(3);

       }

     }

     //-----------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------

     if ((y >= (timePos[1] + 3 * timeButtonSize + 3 * timenumGap)) && (y <= (timePos[1] + 4 *
timeButtonSize + 3 * timenumGap)) && (statusCheck != "paused") ) // LAST ROW

     { // LAST ROW

       if ((x >= (timePos[0])) && (x <= (timePos[0] + timeButtonSize))) // #0

       {

        drawFrame(timePos[0], timePos[1] + 3 * timeButtonSize + 3 * timenumGap, timePos[0] +
timeButtonSize, timePos[1] + 4 * timeButtonSize + 3 * timenumGap);

        updateStr(0);

       }

       if ((x >= timePos[0] + timeButtonSize + timenumGap) && (x <= timePos[0] + 3 * timeButtonSize + 2 *
timenumGap)) // DEL

       {

        drawFrame(timePos[0] + timeButtonSize + timenumGap, timePos[1] + 3 * timeButtonSize + 3 *
timenumGap, timePos[0] + 3 * timeButtonSize + 2 * timenumGap, timePos[1] + 4 * timeButtonSize + 3 *
timenumGap);

        updateStr(99);

       }

     }

     //-----------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------

    }

  }

}
```

```
//----------CUSTOM FUNCTIONS--------------------------------
void updateStr(int val)
{
  if (val == 99)
  {
    if (timerposition == 0)
    {
      timerposition = 0;
    }
    else
    {
      timerposition = timerposition - 1;
      timerStr[timerposition] = String('-');
      timerNumeric[0] = timerStr[0].toInt();
      timerNumeric[1] = timerStr[1].toInt();
      timerNumeric[2] = timerStr[2].toInt();
      timerNumeric[3] = timerStr[3].toInt();
      timerNumeric[4] = timerStr[4].toInt();
      updateTimeDisp(255, 255, 255);
      timeInSeconds = timerNumeric[0] * 3600 + timerNumeric[1] * 60 * 10 + timerNumeric[2] * 60 +
timerNumeric[3] * 10 + timerNumeric[4];
      if (timeInSeconds > 35999)
      {
        timeInSeconds = 35999;
      }
      else
      {
        timeInSeconds = timeInSeconds;
      }
```

```
    }
  }

  else if (timerposition < 5) {
    timerStr[timerposition] = String(val);
    timerposition = timerposition + 1;

    timerNumeric[0] = timerStr[0].toInt();
    timerNumeric[1] = timerStr[1].toInt();
    timerNumeric[2] = timerStr[2].toInt();
    timerNumeric[3] = timerStr[3].toInt();
    timerNumeric[4] = timerStr[4].toInt();
    timeInSeconds = timerNumeric[0] * 3600 + timerNumeric[1] * 60 * 10 + timerNumeric[2] * 60 +
timerNumeric[3] * 10 + timerNumeric[4];
    updateTimeDisp(255, 255, 255);

    if (timeInSeconds > 35999)
    {
      timeInSeconds = 35999;
    }
    else
    {
      timeInSeconds = timeInSeconds;
    }
  }
}

void updateTimeDisp(float color1, float color2, float color3)
{
```

```
  myGLCD.setFont(GroteskBold24x48);
  float fontWidth = myGLCD.getFontXsize();
  float fontHeight = myGLCD.getFontYsize();
  myGLCD.setColor(color1, color2, color3);
  myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);
  myGLCD.print(timerStr[0], timeNumPos[0] - 3.5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(":",    timeNumPos[0] - 2.5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(timerStr[1], timeNumPos[0]  - 1.5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(timerStr[2], timeNumPos[0] - .5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(":",    timeNumPos[0] + .5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(timerStr[3], timeNumPos[0] + 1.5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
  myGLCD.print(timerStr[4], timeNumPos[0] + 2.5 * fontWidth, timeNumPos[1] - .5 * fontHeight);
}

void updateTinyTimeDisp(float color1, float color2, float color3)
{ // Because of the large amount of time needed to draw pixels, a tiny font is used to dynamically keep track of time.
  myGLCD.setFont(TinyFont);
  float fontWidth = myGLCD.getFontXsize();
  float fontHeight = myGLCD.getFontYsize();
  myGLCD.setColor(color1, color2, color3);
  myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);

  if (timerStr[0] != prevtimerStr[0])
  {
    myGLCD.print(timerStr[0], timeNumPos[0] - 3.5 * fontWidth, timeNumPos[1] + 3 * fontHeight);
  }
  if (timerStr[1] != prevtimerStr[1])
  {
```

```
  myGLCD.print(timerStr[1], timeNumPos[0]  - 1.5 * fontWidth, timeNumPos[1] + 3 * fontHeight);
 }
 if (timerStr[2] != prevtimerStr[2])
 {
  myGLCD.print(timerStr[2], timeNumPos[0] - .5 * fontWidth, timeNumPos[1] + 3 * fontHeight);
 }
 if (timerStr[3] != prevtimerStr[3])
 {
  myGLCD.print(timerStr[3], timeNumPos[0] + 1.5 * fontWidth, timeNumPos[1] + 3 * fontHeight);
 }
 if (timerStr[4] != prevtimerStr[4])
 {
  myGLCD.print(timerStr[4], timeNumPos[0] + 2.5 * fontWidth, timeNumPos[1] + 3 * fontHeight);
 }
 prevtimerStr[0] = timerStr[0];
 prevtimerStr[1] = timerStr[1];
 prevtimerStr[2] = timerStr[2];
 prevtimerStr[3] = timerStr[3];
 prevtimerStr[4] = timerStr[4];
}

void cancel()
{
 myGLCD.setFont(TinyFont);
 myGLCD.print("      ",   timeNumPos[0] - 3.5 * 8, timeNumPos[1] + 3 * 8);
 if (statusCheck != "canceled")
 {
  statusCheck = String("canceled");
  digitalWrite(enblPin, LOW); // Disables stepper motor
```

```
    digitalWrite(fanPin, LOW); // Disables DC-fan
    timerNumeric[0] = 0;
    timerNumeric[1] = 0;
    timerNumeric[2] = 0;
    timerNumeric[3] = 0;
    timerNumeric[4] = 0;
    timerStr[0] = String('-');
    timerStr[1] = String('-');
    timerStr[2] = String('-');
    timerStr[3] = String('-');
    timerStr[4] = String('-');
    myGLCD.setBackColor (0, 0, 0);
    myGLCD.setFont(GroteskBold16x32);
    myGLCD.print("       ", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
    myGLCD.setColor(cancelButtonFillColor[0], cancelButtonFillColor[1], cancelButtonFillColor[2]); //TEXT
    myGLCD.print("CANCELED", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
    updateTimeDisp(255, 255, 255);

    timerposition = 0;
    timeInSeconds = 0;

    drawButtons(stdButtonFillColor[0], stdButtonFillColor[1], stdButtonFillColor[2]);
  }
  prevtimerStr[0] = '-';
  prevtimerStr[1] = '-';
  prevtimerStr[2] = '-';
  prevtimerStr[3] = '-';
  prevtimerStr[4] = '-';
}
```

```
void pause()
{
  statusCheck = String("paused");
  digitalWrite(enblPin, LOW); // Disables stepper motor

  myGLCD.setFont(TinyFont);
  myGLCD.setColor(pauseButtonFillColor[0], pauseButtonFillColor[1], pauseButtonFillColor[2]);
  myGLCD.setBackColor(dispButtonFillColor[0], dispButtonFillColor[1], dispButtonFillColor[2]);
  myGLCD.print(timerStr[0], timeNumPos[0] - 3.5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(":",    timeNumPos[0] - 2.5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(timerStr[1], timeNumPos[0]  - 1.5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(timerStr[2], timeNumPos[0] - .5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(":",    timeNumPos[0] + .5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(timerStr[3], timeNumPos[0] + 1.5 * 8, timeNumPos[1] + 3 * 8);
  myGLCD.print(timerStr[4], timeNumPos[0] + 2.5 * 8, timeNumPos[1] + 3 * 8);

  myGLCD.setFont(GroteskBold16x32);
  myGLCD.setBackColor (0, 0, 0);
  myGLCD.print("       ", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
  myGLCD.print("PAUSED", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
}

void steps()
{
  unsigned long trackTimer = millis();
  while ((timeInSeconds > 0) && (statusCheck == "running"))
  {
    while (millis() - trackTimer < 1000) // Run steps for one second, then update timer.
```

```
  {
    myTouch.read();
    x = myTouch.getX();
    y = myTouch.getY();
    if (((x >= cancelButtonPos[0] - .5 * cancelButtonSize[0]) && (x <= cancelButtonPos[0] + .5 *
cancelButtonSize[0])) && ((y >= cancelButtonPos[1] - .5 * cancelButtonSize[1]) && (y <=
cancelButtonPos[1] + .5 * cancelButtonSize[1]))) // CANCEL
    { // CANCEL
      drawFrame(cancelButtonPos[0] - .5 * cancelButtonSize[0] , cancelButtonPos[1] - .5 *
cancelButtonSize[1], cancelButtonPos[0] + .5 * cancelButtonSize[0] , cancelButtonPos[1] + .5 *
cancelButtonSize[1]); // CANCEL
      cancel();
      break;
    }
    else if (((x >= pauseButtonPos[0] - .5 * pauseButtonSize[0]) && (x <= pauseButtonPos[0] + .5 *
pauseButtonSize[0])) && ((y >= pauseButtonPos[1] - .5 * pauseButtonSize[1]) && (y <=
pauseButtonPos[1] + .5 * pauseButtonSize[1]))) // PAUSE
    { //PAUSE
      drawFrame(pauseButtonPos[0] - .5 * pauseButtonSize[0] , pauseButtonPos[1] - .5 *
pauseButtonSize[1], pauseButtonPos[0] + .5 * pauseButtonSize[0] , pauseButtonPos[1] + .5 *
pauseButtonSize[1]); // PAUSE
      pause();
      break;
    }
    else
    {
      digitalWrite(pulPin, HIGH);
      delayMicroseconds(stepper_delay / 2);
      digitalWrite(pulPin, LOW);
      delayMicroseconds(stepper_delay / 2);
    }
  }
```

```
  if (statusCheck == "running")
  {
    trackTimer = millis();
    timeInSeconds = timeInSeconds - 1; // Timer update
    timeHour = timeInSeconds / 3600;
    timeTenMin = (timeInSeconds % 3600) / 600;
    timeMin = (timeInSeconds % 3600) / 60 - timeTenMin * 10;
    timeTenSec = (timeInSeconds % 3600) % 60 / 10;
    timeSec = (timeInSeconds % 3600) % 60 - timeTenSec * 10;
    timerStr[0] = String(timeHour);   //HOUR
    timerStr[1] = String(timeTenMin); //TEN MINUTES
    timerStr[2] = String(timeMin);    //MINUTES
    timerStr[3] = String(timeTenSec); //TEN SECONDS
    timerStr[4] = String(timeSec);    //SECONDS
    updateTinyTimeDisp(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]);
  }
}
if (timeInSeconds == 0 && statusCheck == "running")
{ //Time has successfully run out
  digitalWrite(enblPin, LOW); // Disables stepper motor
  timerposition = 0;
  timerNumeric[0] = 0;
  timerNumeric[1] = 0;
  timerNumeric[2] = 0;
  timerNumeric[3] = 0;
  timerNumeric[4] = 0;
  timerStr[0] = String('-');
  timerStr[1] = String('-');
  timerStr[2] = String('-');
```

```
    timerStr[3] = String('-');
    timerStr[4] = String('-');
    myGLCD.setFont(TinyFont);
    myGLCD.print("      ",    timeNumPos[0] - 3.5 * 8, timeNumPos[1] + 3 * 8);
    myGLCD.setFont(GroteskBold24x48);
    updateTimeDisp(255, 255, 255);

    myGLCD.setBackColor (0, 0, 0);
    myGLCD.setFont(GroteskBold16x32);
    myGLCD.print("       ", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
    myGLCD.setColor(goButtonFillColor[0], goButtonFillColor[1], goButtonFillColor[2]); //TEXT
    myGLCD.print("COMPLETE", goButtonPos[0] - 2 * 24, goButtonPos[1] - 90);
    myGLCD.setFont(GroteskBold24x48);
    drawButtons(stdButtonFillColor[0], stdButtonFillColor[1], stdButtonFillColor[2]);
    statusCheck = String("canceled");
  }
}

// Draw a RED frame while a button is touched
void drawFrame(int x1, int y1, int x2, int y2)
{
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
  while (myTouch.dataAvailable())
    myTouch.read();
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
}
```