```
{\rtf1\ansi\ansicpg1252\cocoartf2511
\cocoatextscaling0\cocoaplatform0{\fonttbl\f0\fswiss\fcharset0
Helvetica;}
{\colortbl;\red255\green255\blue255;}
{\*\expandedcolortbl;;}
\margl1440\margr1440\vieww10800\viewh8400\viewkind0
\pard\tx720\tx1440\tx2160\tx2880\tx3600\tx4320\tx5040\tx5760\tx6480\tx
7200\tx7920\tx8640\pardirnatural\partightenfactor0

\f0\fs24 \cf0 //Add needed libraries-------------------------\
#include <OneWire.h> //Temperature sensor library\
#include <Adafruit_GFX.h>    // Core graphics library\
#include <Adafruit_TFTLCD.h> // Hardware-specific library\
#include <TouchScreen.h>\
\
//Touchscreen
variables-----------------------------------------------------------
-------------------------------------\
int x = 0;\
int y = 0;\
int brewSize = 8;\
int brewStr = 1;\
\
int width = 480;\
int height = 320;\
\
#define YP A3  // must be an analog pin, use "An" notation!\
#define XM A2  // must be an analog pin, use "An" notation!\
#define YM 9   // can be a digital pin\
#define XP 8   // can be a digital pin\
\
#define TS_MINX 150\
#define TS_MINY 120\
#define TS_MAXX 920\
#define TS_MAXY 940\
\
// For better pressure precision, we need to know the resistance\
// between X+ and X- Use any multimeter to read it\
// For the one we're using, its 300 ohms across the X plate\
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);\
\
#define LCD_CS A3\
#define LCD_CD A2\
#define LCD_WR A1\
#define LCD_RD A0\
// optional\
#define LCD_RESET A4\
\
// Assign human-readable names to some common 16-bit color values:\
#define BLACK   0x0000\
```

```
#define BLUE    0x001F\
#define RED     0xF800\
#define GREEN   0x07E0\
#define CYAN    0x07FF\
#define MAGENTA 0xF81F\
#define YELLOW  0xFFE0\
#define WHITE   0xFFFF\
#define BROWN   0XB304\
\
\
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);\
\
#define BOXSIZE 40\
#define PENRADIUS 3\
\
//other variables\
#define MINPRESSURE 10\
#define MAXPRESSURE 1000\
\
#if defined(__SAM3X8E__)\
#undef __FlashStringHelper::F(string_literal)\
#define F(string_literal) string_literal\
#endif\
\
//User preference
variables----------------------------------------------------------
--------------------------------\
int desiredTemp = 95;\
int pressStrength = 1;\
int servingSize = 8;\
int pressTime = 1;\
int firstTime = 1;\
int Selection = 1, incomingByte;\
\
\
// State
variables----------------------------------------------------------
-----------------------------------------\
char machineState = 'S';\
int waterTemp = 0; //Water temperature in celsius\
int setupDone = 0; //should either be 1(yes) or 0 (no), initially set
to off\
\
\
//Heater & temperature sensor
variables----------------------------------------------------------
----------------------\
int heater = 11; //heater pinOut\
int tempSensor = 7; //tempSensor pinOut\
OneWire  ds(tempSensor);\
```

```
\
//Press
variables-----------------------------------------------------------------
-----------------------------------------\
int pressPull = 9; //linear actuator pulse pinOut\
int pressDir = 8; //linear actuator direction pinOut\
\
//Pump
variables-----------------------------------------------------------------
-----------------------------------------\
int pump = 13; //water pump pinOut\
int dispenseTime = 0;\
int pumpVal = 0;\
\
//Grounds dispenser
variables-----------------------------------------------------------------
---------------------------------\
int dispensor = 12; //dispensor pinOut\
int vibrator = 10;  //vibrator pinOut\
int numOfSteps = 10;\
\
\
\
void setup(void) \{\
  Serial.begin(9600);\
\
\
\
//  //initialize\
//  tft.reset();\
//\
//  uint16_t identifier = tft.readID();\
//\
//  if (identifier == 0x9325)\
//  \{\
//    Serial.println(F("Found ILI9325 LCD driver"));\
//  \}\
//  else if (identifier == 0x9328)\
//  \{\
//    Serial.println(F("Found ILI9328 LCD driver"));\
//  \}\
//  else if (identifier == 0x7575)\
//  \{\
//    Serial.println(F("Found HX8347G LCD driver"));\
//  \}\
//  else if (identifier == 0x9341)\
//  \{\
//    Serial.println(F("Found ILI9341 LCD driver"));\
//  \}\
//  else if (identifier == 0x8357)\
```

```
//  \{\
//    Serial.println(F("Found HX8357D LCD driver"));\
//  \}\
//  else\
//  \{\
//    Serial.print(F("Unknown LCD driver chip: "));\
//    Serial.println(identifier, HEX);\
//    Serial.println(F("If using the Adafruit 2.8\\" TFT Arduino
shield, the line:"));\
//    Serial.println(F("  #define USE_ADAFRUIT_SHIELD_PINOUT"));\
//    Serial.println(F("should appear in the library header
(Adafruit_TFT.h)."));\
//    Serial.println(F("If using the breakout board, it should NOT be
#defined!"));\
//    Serial.println(F("Also if using the breakout, double-check that
all wiring"));\
//    Serial.println(F("matches the tutorial."));\
//    return;\
//  \}\
//\
//  tft.begin(identifier);\
//  tft.setRotation(1);\
//  tft.fillScreen(BLUE);\
  //drawScreen();\
  //tft.stroke(WHITE);\
\
\
  //tft.fillRect(20, 20,200 ,80, BROWN);\
\
  int firstTime=1;\
  int Selection = 1;\
\
  int l1height = 50;\
  //Set pins\
  pinMode(heater, OUTPUT);\
  pinMode(pump, OUTPUT);\
  pinMode(dispensor, OUTPUT);\
  pinMode(vibrator, OUTPUT);\
  pinMode(pressPull, OUTPUT);\
  pinMode(pressDir, OUTPUT);\
  pinMode(13, OUTPUT);\
\
\}\
\
// End of setup\
// Beginning of main loop\
//
**********************************************************************
**********************************************************************
************************************************\
```

```
//
**********************************************************************
**********************************************************************
************************************************\
//
**********************************************************************
**********************************************************************
************************************************\
//
**********************************************************************
**********************************************************************
************************************************\
void loop(void) \{\
  // System decides what to do based on state variable. State variable
is initialized to "User Setup"\
\
  //updates Display based on state variables\
  //Serial.println(machineState);\
  brewState:\
\
  //User
Stetup-----------------------------------------------------------------
---------------\
  if (machineState == 'S') \{\
    //Serial.print(firstTime);\
    if (firstTime == 1)\
    \{\
      firstTime = 0;\
      if (Selection == 1)\
      \{ \
        Serial.print("Type in a Brew Size (8,16,24 oz.): ");\
      \}\
\
      else if (Selection ==2)\
      \{\
        Serial.print("Type in a Brew Strength (1-mild, 2-regular, 3-
strong): ");\
      \}\
\
      else if (Selection == 3)\
      \{\
        Serial.print("Type in a Press Time in seconds(1-15s, 2-30s,
3-45s): ");\
      \}\
    \}\
    if (Serial.available() > 0) \
    \{\
      // read the incoming byte:\
      incomingByte =  Serial.readString().toInt();\
  \
```

```
      if (Selection == 1)\
      \{\
        servingSize = (int) incomingByte;\
        servingSize = max(8, servingSize);\
        servingSize = min(24, servingSize);\
        Selection = 2;\
        firstTime=1;\
        Serial.println(incomingByte);\
        \
      \}\
\
      else if (Selection == 2)\
      \{\
        pressStrength = (int) incomingByte;\
        pressStrength = max(1, pressStrength);\
        pressStrength = min(3, pressStrength);\
        Selection = 3;\
        firstTime=1;\
        Serial.println(incomingByte);\
      \}\
      else if (Selection == 3)\
      \{\
        pressTime = (int) incomingByte;\
        pressTime = max(1, pressTime);\
        pressTime = min(3, pressTime);\
        machineState = 'H';\
        Serial.println(incomingByte);\
      \}\
    \}\
    //TSPoint p = ts.getPoint();\
    //digitalWrite(13, LOW);\
    //pinMode(XM, OUTPUT);\
    //pinMode(YP, OUTPUT);\
     \
    //Serial.println(F("Test!"));\
\
//    if (p.z > MINPRESSURE && p.z < MAXPRESSURE) \{\
//\
//      p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.height());\
//      p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.width());\
//\
//      double tempPy = map(p.x, 0, 320, 320, 0);\
//      double tempPx = map(p.y, 0, 480, 0, 480);\
//\
//      p.x = tempPx;\
//      p.y = tempPy;\
//\
//      //Serial.print("Px: ");\
//      //Serial.println(p.x);\
//\
```

```
//     // Serial.print("Py: ");\
//      //Serial.println(p.y);\
//\
//      double px = p.x;\
//      double py = p.y;\
//      //drawScreen();\
//      //checkButtonPressed(px, py);\
//      //tft.fillRect(0, BOXSIZE, tft.width(), tft.height()-BOXSIZE,
BLACK);\
//      //tft.fillScreen(BLACK);\
//\
//    \} \
  \}\
\
  //Heat
Water-----------------------------------------------------------------
------------------\
  else if (machineState == 'H')\
  \{\
    //updateDisplay('H');\
    heatWater();\
\
    //State change determined inside heatWater() function unlike other
functions\
  \}\
\
  //Move Grounds &
Water-----------------------------------------------------------------
----------\
  else if (machineState == 'G')\
  \{\
    updateDisplay('G');\
    dispenseGroundsandWater();\
    machineState = 'P';\
    //tft.fillScreen(WHITE);\
  \}\
\
\
  //Press
coffee----------------------------------------------------------------
--------------------\
  else if (machineState == 'P')\
  \{\
    //updateDisplay('P');\
    pressCoffee();\
    machineState = 'C';\
    //tft.fillScreen(WHITE);\
  \}\
  // Coffee is
done!-----------------------------------------------------------------
```

```
----------------\
  else if (machineState == 'C')\
  \{\
     updateDisplay('C');\
    TSPoint p = ts.getPoint();\
    digitalWrite(13, LOW);\
    pinMode(XM, OUTPUT);\
    pinMode(YP, OUTPUT);\
\
    if (p.z > MINPRESSURE && p.z < MAXPRESSURE) \{\
\
      p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.height());\
      p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.width());\
\
      double tempPy = map(p.x, 0, 320, 320, 0);\
      double tempPx = map(p.y, 0, 480, 0, 480);\
\
      p.x = tempPx;\
      p.y = tempPy;\
\
      Serial.print("Px: ");\
      Serial.println(p.x);\
\
      Serial.print("Py: ");\
      Serial.println(p.y);\
\
      double px = p.x;\
      double py = p.y;\
      checkOk(px, py);\
      //tft.fillRect(0, BOXSIZE, tft.width(), tft.height()-BOXSIZE,
BLACK);\
      //tft.fillScreen(BLACK);\
      \
      //machineState = 'H';\
\
    \}\
   \
\
  \}\
\
  //Device is set to
off-------------------------------------------------------------------
------------\
  else if (machineState == 'O')\
  \{\
    //...\
  \}\
\
\}\
// End of main loop\
```

```
//
**********************************************************************
**********************************************************************
************************************************\
//
**********************************************************************
**********************************************************************
************************************************\
//
**********************************************************************
**********************************************************************
************************************************\
//
**********************************************************************
**********************************************************************
************************************************\
\
//LCD User Display
Functions-------------------------------------------------------\
\
void updateDisplay(char displayText)\
\{\
  tft.setTextColor(BLACK);\
  tft.setTextSize(4);\
  tft.setCursor(60, 50);\
  if (displayText == 'H')\
    tft.print("Status: Heating");\
  else  if (displayText == 'G')\
    tft.print("Status: Preparing");\
  else  if (displayText == 'P')\
    tft.print("Status: Pressing");\
  else  if (displayText == 'P') \{\
    tft.print("Status: Complete");\
    tft.setTextColor(WHITE);\
    tft.fillRect(140, 240, 200 , 50, BROWN);\
    tft.setTextSize(4);\
    tft.setCursor(218, 250);\
    tft.print("OK");\
  \}\
\}\
\
\
//User Setup
Functions-------------------------------------------------------\
\
\
void drawScreen() \{\
\
  drawBrewSize();\
  drawBrewStrength();\
```

```
  drawPressTime();\
  drawStart();\
\
\
\}\
\
void drawStart() \{\
  tft.setTextColor(WHITE);\
  tft.fillRect(140, 240, 200 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(182, 250);\
  tft.print("Start");\
\}\
void drawPressTime() \{\
  int l1height = 50;\
  int l3increment = 120;\
  tft.setTextColor(BLACK);\
  tft.setTextSize(3);\
  tft.setCursor(30, l1height + l3increment);\
  tft.print("Press Time:");\
\
  //Left Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(220, 40 + l3increment, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(230, l1height + l3increment);\
  tft.print("<");\
\
  // Brew Size\
  tft.fillRect(270, 40 + l3increment, 100 , 50, WHITE);\
  tft.setTextColor(BLACK);\
  tft.setCursor(280, l1height + l3increment);\
  tft.setTextSize(3);\
  tft.print(pressTime);\
  tft.print(" s ");\
\
  //Right Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(370, 40 + l3increment, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(385, l1height + l3increment);\
  tft.print(">");\
\
\
\}\
\
void drawBrewStrength() \{\
  int l1height = 50;\
  int l2increment = 60;\
  tft.setTextColor(BLACK);\
```

```
  tft.setTextSize(3);\
  tft.setCursor(30, l1height + l2increment);\
  tft.print("Strength:");\
\
  //Left Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(220, 40 + l2increment, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(230, l1height + l2increment);\
  tft.print("<");\
\
  // Brew Size\
  tft.fillRect(270, 40 + l2increment, 100 , 50, WHITE);\
  tft.setTextColor(BLACK);\
  tft.setCursor(280, l1height + l2increment);\
  tft.setTextSize(3);\
  if (brewStr == 1)\
    tft.print("Mild");\
  else if (brewStr == 2)\
    tft.print("Reg");\
  else if (brewStr == 3)\
    tft.print("Str");\
\
\
  //Right Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(370, 40 + l2increment, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(385, l1height + l2increment);\
  tft.print(">");\
\
\
\}\
void drawBrewSize() \{\
  //tft.fillScreen(WHITE);\
  int l1height = 50;\
  tft.setTextColor(BLACK);\
  tft.setRotation(1);\
  tft.setTextSize(3);\
  tft.setCursor(30, l1height);\
  tft.print("Brew Size:");\
\
\
  //Left Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(220, 40, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(230, l1height);\
  tft.print("<");\
\
```

```
  // Brew Size\
  tft.fillRect(270, 40, 100 , 50, WHITE);\
  tft.setTextColor(BLACK);\
  tft.setCursor(280, l1height);\
  tft.setTextSize(3);\
  tft.print(brewSize);\
  tft.print("oz.");\
\
\
  //Right Arrow\
  tft.setTextColor(WHITE);\
  tft.fillRect(370, 40, 50 , 50, BROWN);\
  tft.setTextSize(4);\
  tft.setCursor(385, l1height);\
  tft.print(">");\
\
\
\}\
double checkButtonPressed(double px, double py) \{\
\
\
  // Brew size buttons\
  if (px > 370 && px < 370 + 50 && py > 40 && py < 40 + 50)\
  \{\
    brewSize = max(brewSize + 8, 8);\
    brewSize = min(brewSize, 24);\
  \}\
\
  else if (px > 220 && px < 220 + 50 && py > 40 && py < 40 + 50)\
  \{\
    brewSize = max(brewSize - 8, 8);\
    brewSize = min(brewSize, 24);\
  \}\
  // Brew Strength\
  else if (px > 370 && px < 370 + 50 && py > 100 && py < 100 + 50)\
  \{\
    brewStr = max(brewStr + 1, 1);\
    brewStr = min(brewStr, 3);\
  \}\
  else if (px > 220 && px < 220 + 50 && py > 100 && py < 100 + 50)\
  \{\
    brewStr = max(brewStr - 1, 1);\
    brewStr = min(brewStr, 3);\
  \}\
  // Press Time\
  else if (px > 370 && px < 370 + 50 && py > 160 && py < 160 + 50)\
  \{\
    pressTime = max(pressTime + 15, 15);\
    pressTime = min(pressTime, 45);\
  \}\
```

```
  else if (px > 220 && px < 220 + 50 && py > 160 && py < 160 + 50)\
  \{\
    pressTime = max(pressTime - 15, 15);\
    pressTime = min(pressTime, 45);\
  \}\
  // Start\
  else if (px > 140 && px < 140 + 200 && py > 240 && py < 240 + 50)\
  \{\
    machineState = 'H';\
    tft.fillScreen(WHITE);\
    \
    if (brewStr == 1)\
      numOfSteps = 7;\
    else if (brewStr == 2)\
      numOfSteps = 10;\
    else if (brewStr == 3)\
      numOfSteps = 14;\
\
    if (brewSize == 8)\
      pumpVal = 5*1000;\
    else if (brewSize == 16)\
      numOfSteps = 15*1000;\
    else if (brewSize == 24)\
      numOfSteps = 25*1000;\
    \
  \}\
\
  delay(200);\
  return px;\
\
\}\
\
void checkOk(double px, double py) \{\
  if (px > 140 && px < 140 + 200 && py > 240 && py < 240 + 50)\
  \{\
    machineState = 'S';\
    tft.fillScreen(WHITE);\
  \}\
\
  delay(200);\
\}\
\
//Heater
Functions-----------------------------------------------------------
--\
void heatWater()\
\{\
  //digitalWrite(heater, HIGH); \
  waterTemp = getTemp();\
  pinMode(heater, OUTPUT);\
```

```
  Serial.print(waterTemp);\
  if (waterTemp >= desiredTemp)\
  \{\
    machineState = 'G'; //changes state\
    digitalWrite(heater, LOW); //turns heater OFF\
    //tft.fillScreen(WHITE);\
  \}\
\
\
  else\
  \{\
    digitalWrite(heater, HIGH); //turns heater ON\
    //tft.setTextColor(BLACK);\
    //tft.setTextSize(3);\
    //tft.setCursor(70, 170);\
    //tft.print("Current Temp: ");\
    //tft.print(waterTemp);\
    //tft.print(" C");\
  \}\
\
\
\
\
\}\
float getTemp() \{\
\
  byte i;\
  byte present = 0;\
  byte type_s;\
  byte data[12];\
  byte addr[8];\
  float celsius, fahrenheit, heaterCutTemp = 95;\
\
  if ( !ds.search(addr)) \{\
    //Serial.println("No more addresses.");\
    //  Serial.println();\
    ds.reset_search();\
    delay(250);\
    return;\
  \}\
\
\
  if (OneWire::crc8(addr, 7) != addr[7]) \{\
    Serial.println("CRC is not valid!");\
    return;\
  \}\
  //Serial.println();\
\
  // the first ROM byte indicates which chip\
  switch (addr[0]) \{\
```

```
    case 0x10:\
      //Serial.println("  Chip = DS18S20");  // or old DS1820\
      type_s = 1;\
      break;\
    case 0x28:\
      //Serial.println("  Chip = DS18B20");\
      type_s = 0;\
      break;\
    case 0x22:\
      //Serial.println("  Chip = DS1822");\
      type_s = 0;\
      break;\
    default:\
      Serial.println("Device is not a DS18x20 family device.");\
      return;\
  \}\
\
  ds.reset();\
  ds.select(addr);\
  ds.write(0x44);        // start conversion, use ds.write(0x44,1)
with parasite power on at the end\
\
  delay(750);     // maybe 750ms is enough, maybe not\
  // we might do a ds.depower() here, but the reset will take care of
it.\
\
  present = ds.reset();\
  ds.select(addr);\
  ds.write(0xBE);         // Read Scratchpad\
\
  for ( i = 0; i < 9; i++) \{           // we need 9 bytes\
    data[i] = ds.read();\
  \}\
  Serial.println();\
\
  // Convert the data to actual temperature\
  // because the result is a 16 bit signed integer, it should\
  // be stored to an "int16_t" type, which is always 16 bits\
  // even when compiled on a 32 bit processor.\
  int16_t raw = (data[1] << 8) | data[0];\
  if (type_s) \{\
    raw = raw << 3; // 9 bit resolution default\
    if (data[7] == 0x10) \{\
      // "count remain" gives full 12 bit resolution\
      raw = (raw & 0xFFF0) + 12 - data[6];\
    \}\
  \} else \{\
    byte cfg = (data[4] & 0x60);\
    // at lower res, the low bits are undefined, so let's zero them\
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms\
```

```
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms\
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms\
    //// default is 12 bit resolution, 750 ms conversion time\
  \}\
  celsius = (float)raw / 16.0;\
  Serial.print(celsius);\
  return celsius;\
\
\}\
\
\
\
\
//Grounds & pump
Functions------------------------------------------------------------
----------------------------------------------------\
\
void dispenseGroundsandWater() \{\
\
  pinMode(pump,OUTPUT);\
  pinMode(dispensor, OUTPUT);\
  pinMode(vibrator, OUTPUT);\
\
  if (pressStrength == 1)\
    numOfSteps = 6 + 2 * (servingSize/2);\
  else if (pressStrength == 2)\
    numOfSteps = 7 + 2 * (servingSize/2);\
  else if (pressStrength == 3)\
    numOfSteps = 9 + 2 * (servingSize/2);\
\
    \
  for (int i = 0; i < numOfSteps; i++)\
  \{\
    digitalWrite(dispensor, HIGH);\
    digitalWrite(pump, HIGH);\
    delay(5000);\
    digitalWrite(pump, LOW);\
    delay(1000);\
    digitalWrite(dispensor, LOW);\
    digitalWrite(pump, LOW);\
    \
    digitalWrite(vibrator, HIGH);\
    delay(1200);\
    digitalWrite(vibrator, LOW);\
\
  \}\
\
  digitalWrite(pump, LOW);\
  digitalWrite(dispensor, LOW);\
\}\
```

```
\
\
void runPump() \{\
    if (servingSize == 24)\
    pumpVal = 5000;\
  else if (servingSize == 16)\
    pumpVal = 12500;\
  else if (servingSize == 8)\
    pumpVal = 20000;\
\
  digitalWrite(pump, HIGH);\
  delay(pumpVal);\
  digitalWrite(pump, LOW);\
\}\
\
\
\
void determineTime()\
\{\
  //...\
\}\
\
void dispenseGrounds()\
\{\
  //...\
\}\
\
\
void pumpWater()\
\{\
  //....\
\}\
// Press
Functions----------------------------------------------------------------
------------------------------------------------------------------------
-----\
\
void pressCoffee() \{\
  pinMode(pressDir, OUTPUT);\
  pinMode(pressPull, OUTPUT);\
  int pressDistance = 0;\
  int holdPressTime = 0;\
\
  Serial.println(pressTime);\
\
  if (pressStrength == 3)\
  \{\
    pressDistance = 10; //9 inches pressed\
    holdPressTime = 90; // hold press for 120 seconds (2 minutes)\
  \}\
```

```
  else if (pressStrength == 2)\
  \{\
    pressDistance = 9.5; // 8.5 inches pressed\
    holdPressTime = 60; // hold press for 100 seconds (1 minute 40
seconds)\
  \}\
\
  else if (pressStrength == 1)\
  \{\
    pressDistance = 9; // 8 inches pressed\
    holdPressTime = 45; // hold press for 90 seconds (1 minute 30
seconds)\
  \}\
\
  float dpr = .31496; //(distance per revolution) linear distance
(inches) press travelled per one full rotation\
  int spr = 400; //(steps per revolution) number of steps per one full
rotation of stepper motor (step angle 0.9 deg)\
  int numOfSteps = (int) (pressDistance / dpr) * spr;\
\
  int pressSpeed = 1 / 2; // inches travelled per second\
  int stepLength = (int) dpr * 1000000 / (2 * spr * pressSpeed); //
time length of each step\
\
\
  numOfSteps =27000;\
  stepLength = 400;\
\
  Serial.println("Pressing coffee");\
  Serial.print("numOfSteps: ");\
  Serial.println(numOfSteps);\
    Serial.print("stepLength: ");\
  Serial.println(stepLength);\
\
  \
  // Press down coffee in carafe\
  digitalWrite(pressDir, LOW);\
  for (int i = 0; i < numOfSteps; i++)\
  \{\
    digitalWrite(pressPull, HIGH);\
    delayMicroseconds(stepLength);\
    digitalWrite(pressPull, LOW);\
    delayMicroseconds(stepLength);\
    //Serial.println(i);\
  \}\
\
  // Hold press position for time length specified by"holdPressTime"\
  delay(holdPressTime * 1000);\
  \
  //Retract press from carafe\
```

```
  digitalWrite(pressDir, HIGH);\
  for (int i = 0; i < numOfSteps; i++)\
  \{\
    digitalWrite(pressPull, HIGH);\
    delayMicroseconds(stepLength);\
    digitalWrite(pressPull, LOW);\
    delayMicroseconds(stepLength);\
  \}\
\
\
\}}
```