HOME BODY
2/22/22

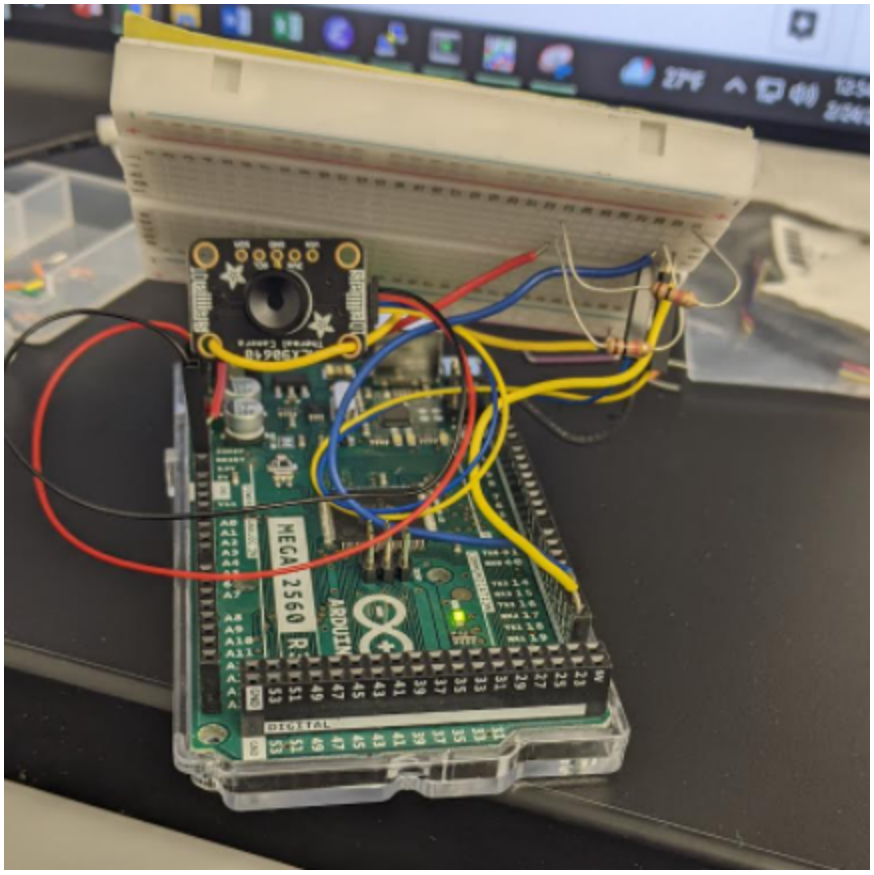
Sensor Update

Status

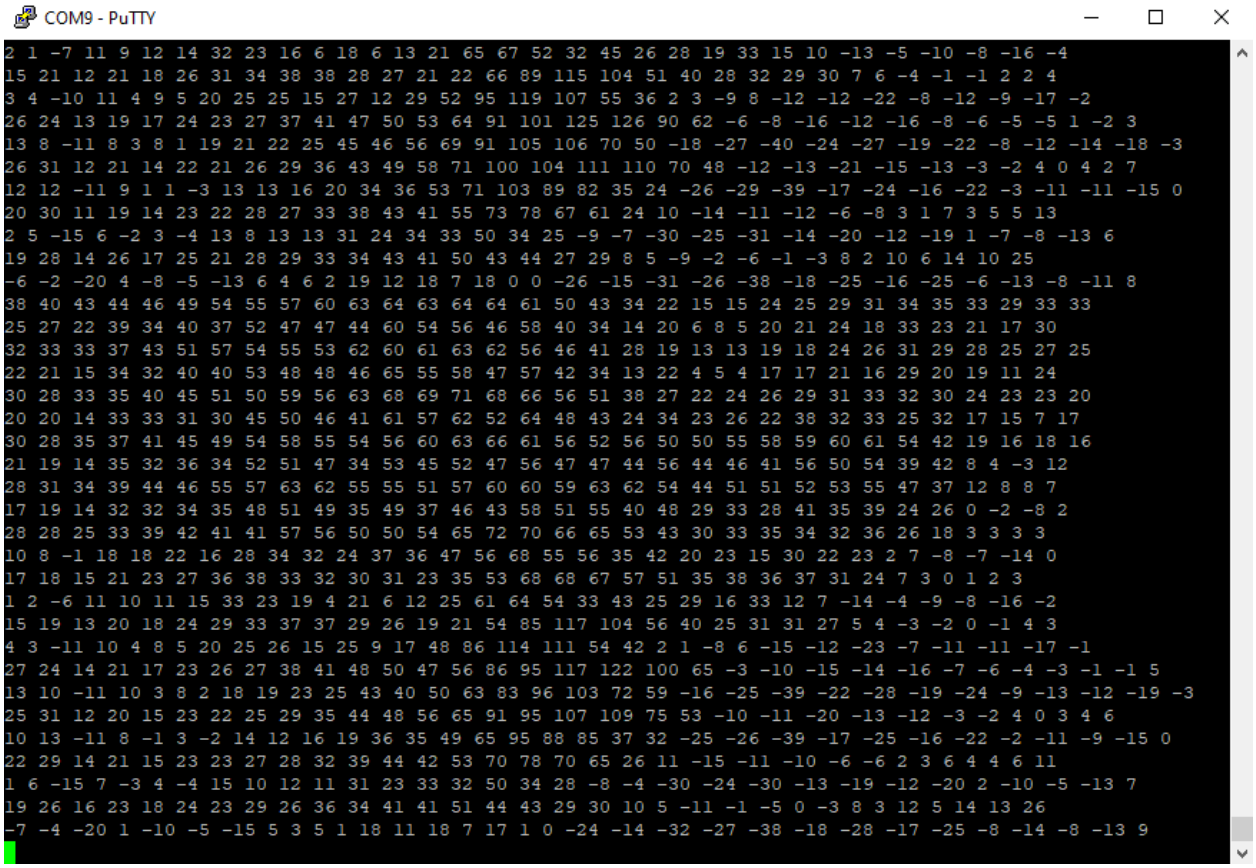
- The MLX9640 IR sensor has been connected to the arduino mega 2560 via i2c protocol.
- Rudimentary software has been written to interface the board back to a host computer over a serial (COM) port.
- The data from the COM port is logged by PuTTY and a GNUPlot script checks this log file to update a heatmap plot in soft real time.
- The sensor is capturing images and converting readings to celsius temperatures. The accuracy of this conversion needs to be tested.
- The refresh rate and interlacing (the reason for the chessboard style images) needs to be improved.
- Current memory usage on board: Sketch uses 6820 bytes (2%) of program storage space. Maximum is 253952 bytes.Global variables use 588 bytes (7%) of dynamic memory, leaving 7604 bytes for local variables. Maximum is 8192 bytes. If I discover myself to be a good C programmer there is a chance that we can squeeze this onto the wifi uno board (using less power and having onboard wifi!)

Board Setup Picture

The setup of the board was described above in status but is shown below. Blue wire is SDA (data for i2c protocol) yellow wire is SCL (clock for i2c protocol) These are connected to Vdd (supply voltage) via 2 10k ohm pullup resistors. This is a crude and temporary configuration while the enclosure is designed and printed.


MEGA 2560 R3
ARDUINO
DIGITAL

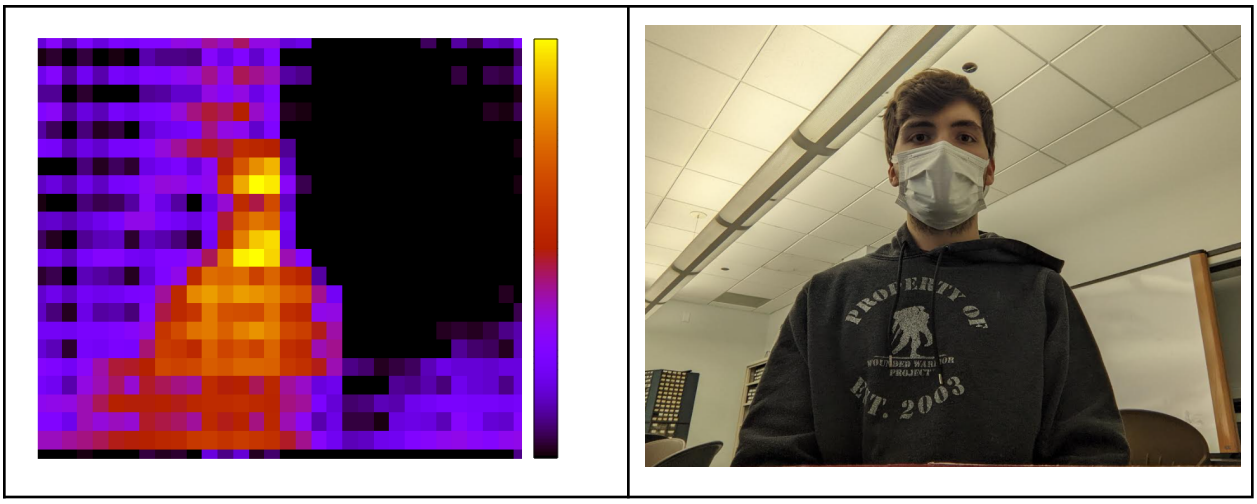
Using PuTTY to get sensor data to host computer



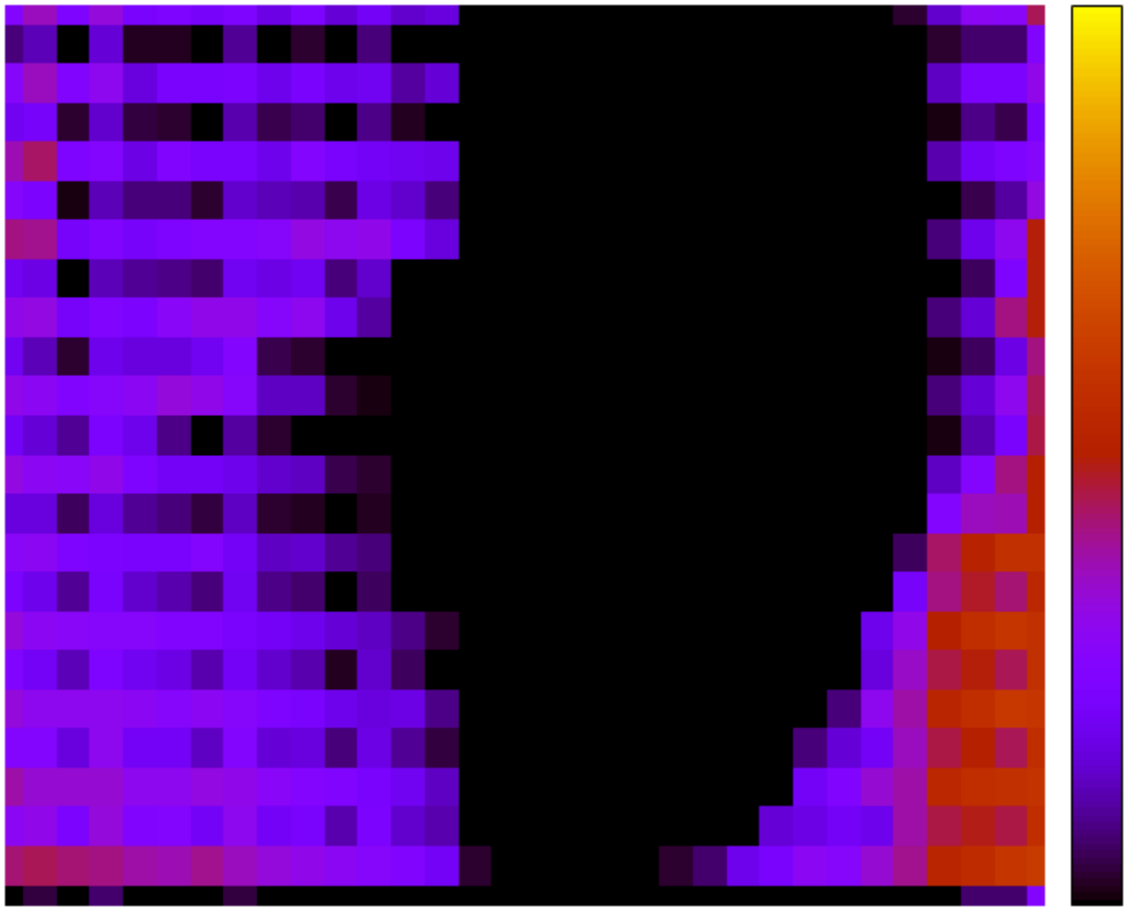
While connected to a host computer via a USB-B cable, serial protocol currently at 115200 buad (bit/s). This can be seen in the source code below where we have:
Serial.begin(115200);
The image itself displays the raw data being sent from the sensor.

Sample Images

Hello world image of me sitting in front of the sensor. Side by side with an image taken on my phone of the same view. This image shows the interlace issue (checkerboard pattern). Yellow ~ 100C, black ~0C.



If I step outside the FOV of the sensor it takes the following image (you can actually still see some of my heat on the right) This image also shows the ‘hot corners’ where the heat of the device itself registers. When the sensor is turned on there is a linear adjustment (subtraction) done to compensate for this. As can be seen, it is not perfect.



Sensor code (C Language)

```
/*
  HOME BODY
  srd
  Read the temperature pixels from the MLX90640 IR array
  this code is a rewrite driver for the MLX90640 to allow working on arudino.
  Hardware Connections:
  (https://www.sparkfun.com/products/14425)

*/
#include <Wire.h>
#include <avr/pgmspace.h>


#include <math.h>

//I2C_BUFFER_LENGTH is defined in Wire.H

//#define I2C_BUFFER_LENGTH BUFFER_LENGTH
#define I2C_BUFFER_LENGTH 32


/*
  //Returns true if the MLX90640 is detected on the I2C bus
  boolean isConnected()
  {
  Wire.beginTransmission((uint8_t)MLX90640_address);
  if (Wire.endTransmission() != 0)
  return (false); //Sensor did not ACK
  return (true);
  }
*/
// this function is registered as an event, see setup()
void receiveEvent(int howMany) {
  while (1 < Wire.available()) { // loop through all but the last
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);         // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);         // print the integer
}
```

```
int MLX90640_I2CWrite(uint8_t _deviceAddress, unsigned int writeAddress, uint16_t data)
{
  Wire.beginTransmission((uint8_t)_deviceAddress);
  Wire.write(writeAddress >> 8); //MSB
  Wire.write(writeAddress & 0xFF); //LSB
  Wire.write(data >> 8); //MSB
  Wire.write(data & 0xFF); //LSB

  if (Wire.endTransmission() != 0)
  {
    //Sensor did not ACK
    Serial.println("Error: Sensor did not ack");
    return (-1);
  }

  uint16_t dataCheck;
  int   MLX90640_I2CRead(_deviceAddress, writeAddress, 1, &dataCheck);
  if (dataCheck != data)
  {
    //Serial.println("The write request didn't stick");
    return -2;
  }

  return (0); //Success
}




int MLX90640_I2CRead(uint8_t _deviceAddress, unsigned int startAddress, unsigned int
nWordsRead, uint16_t *data)
{

  //Caller passes number of 'unsigned ints to read', increase this to 'bytes to read'
  uint16_t bytesRemaining = nWordsRead * 2;

  //It doesn't look like sequential read works. Do we need to re-issue the address command
each time?

  uint16_t dataSpot = 0; //Start at beginning of array

  //Setup a series of chunked I2C_BUFFER_LENGTH byte reads
  while (bytesRemaining > 0)
  {
    Wire.beginTransmission(_deviceAddress);
    Wire.write(startAddress >> 8); //MSB
    Wire.write(startAddress & 0xFF); //LSB
```

```
    if (Wire.endTransmission(false) != 0) //Do not release bus
    {
      Serial.println("No ack read");
      return (0); //Sensor did not ACK
    }

    uint16_t numberOfBytesToRead = bytesRemaining;
    if (numberOfBytesToRead > I2C_BUFFER_LENGTH) numberOfBytesToRead =
I2C_BUFFER_LENGTH;

    Wire.requestFrom((uint8_t)_deviceAddress, numberOfBytesToRead);
    if (Wire.available())
    {
      for (uint16_t x = 0 ; x < numberOfBytesToRead / 2; x++)
      {
        //Store data into array
        data[dataSpot] = Wire.read() << 8; //MSB
        data[dataSpot] |= Wire.read(); //LSB

        dataSpot++;
      }
    }

    bytesRemaining -= numberOfBytesToRead;

    startAddress += numberOfBytesToRead / 2;
  }

  return (0); //Success
}

int MLX90640_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
{
  return MLX90640_I2CRead(slaveAddr, 0x2400, 832, eeData);
}



int MLX90640_GetFrameData(uint8_t slaveAddr, uint16_t *frameData)
{
  uint16_t dataReady = 1;
  uint16_t controlRegister1;
  uint16_t statusRegister;
  int error = 1;
  uint8_t cnt = 0;
  Serial.println(".");
  dataReady = 0;
  while (dataReady == 0)
  {
    error = MLX90640_I2CRead(slaveAddr, 0x8000, 1, &statusRegister);
```

```
    if (error != 0)
    {
      return error;
    }
    dataReady = statusRegister & 0x0008;
  }

  while (dataReady != 0 && cnt < 5)
  {
    error = MLX90640_I2CWrite(slaveAddr, 0x8000, 0x0030);
    if (error == -1)
    {
      return error;
    }

    error = MLX90640_I2CRead(slaveAddr, 0x0400, 832, frameData);
    if (error != 0)
    {
      return error;
    }

    error = MLX90640_I2CRead(slaveAddr, 0x8000, 1, &statusRegister);
    if (error != 0)
    {
      return error;
    }
    dataReady = statusRegister & 0x0008;
    cnt = cnt + 1;
  }

  if (cnt > 4)
  {
    return -8;
  }

  error = MLX90640_I2CRead(slaveAddr, 0x800D, 1, &controlRegister1);
  frameData[832] = controlRegister1;
  frameData[833] = statusRegister & 0x0001;

  if (error != 0)
  {
    return error;
  }

  return frameData[10];
}
int MLX90640_GetCurResolution(uint8_t slaveAddr)
{
  uint16_t controlRegister1;
  int resolutionRAM;
```

```
  int error;

  error = MLX90640_I2CRead(slaveAddr, 0x800D, 1, &controlRegister1);
  if (error != 0)
  {
    return error;
  }
  resolutionRAM = (controlRegister1 & 0x0C00) >> 10;

  return resolutionRAM;
}

int MLX90640_GetCurMode(uint8_t slaveAddr)
{
  uint16_t controlRegister1;
  int modeRAM;
  int error;

  error = MLX90640_I2CRead(slaveAddr, 0x800D, 1, &controlRegister1);
  if (error != 0)
  {
    return error;
  }
  modeRAM = (controlRegister1 & 0x1000) >> 12;

  return modeRAM;
}

int MLX90640_GetRefreshRate(uint8_t slaveAddr)
{
  uint16_t controlRegister1;
  int refreshRate;
  int error;

  error = MLX90640_I2CRead(slaveAddr, 0x800D, 1, &controlRegister1);
  if (error != 0)
  {
    return error;
  }
  refreshRate = (controlRegister1 & 0x0380) >> 7;

  return refreshRate;
}
void printBits(byte myByte) {
  for (byte mask = 0x80; mask; mask >>= 1) {
    if (mask  & myByte)
      Serial.print('1');
    else
      Serial.print('0');
  }
```

```
}



#define MLX90640_address 0x33 //Default 7-bit unshifted address of the MLX90640
byte status = 0; //we use for status
static byte mlx90640To[10];

//below lines change important settings of sensor
#define continuousmode true //true is default when sensor is bought, however we want step mode. this is checked by the code and only written to if it is different than value here. it is here for experimentation.
#define hzMode 2           //0=0.5hz,1=1hz,2=2hz,3=4hz,4=8hz,5=16hz,6=32hz,7=64hz
#define adSensorResolution 0 //0=16bit,it 1=17bit, 2=18bit, 3=19b
////************************ calibration store. first we run program that dumps the data, and then we cut and paste it here in bracekts below*********
const PROGMEM uint16_t factoryCalData[] =  {
  //data goes here
  171, 10651, 0, 8289, 5, 800, 992, 6670, 41522, 389, 1165, 0, 6401, 0, 0, 48691,
  16912, 65471, 514, 514, 61954, 58098, 53729, 41152, 65038, 65039, 61168, 61424, 61442, 61700, 61957, 58116,
  31142, 11880, 65500, 8720, 13106, 8755, 65297, 48094, 60892, 255, 8721, 8738, 8755, 4386, 61184, 52446,
  5946, 12188, 8532, 40315, 21555, 63952, 27241, 26211, 9059, 62547, 5052, 829, 60416, 62208, 40645, 9560,
  1998, 1954, 6416, 63568, 2142, 144, 6206, 61534, 62, 64, 8094, 64430, 64622, 64638, 5200, 62542,
  910, 64336, 5152, 60430, 64448, 64446, 5072, 62366, 63568, 62480, 4128, 60654, 64576, 61680, 5074, 61120,
  2050, 2000, 63710, 4096, 3088, 94, 64510, 4112, 2, 1070, 65390, 6000, 78, 1102, 63550, 4112,
  1890, 816, 64510, 4048, 914, 896, 65454, 4944, 1058, 992, 65534, 4256, 3074, 192, 63504, 3840,
  1086, 1952, 5232, 63422, 3006, 64544, 7072, 62414, 1982, 2014, 7968, 63358, 2032, 30, 5136, 61470,
  65374, 64286, 6096, 61358, 64368, 64336, 5008, 61278, 65520, 62384, 5008, 59504, 912, 62560, 5170, 60288,
  2192, 2000, 62654, 4064, 4066, 64622, 64510, 4096, 2032, 1054, 894, 5054, 2112, 110, 63614, 4192,
  1952, 878, 63550, 4080, 65474, 65454, 64478, 4000, 1060, 65520, 65502, 3248, 4050, 160, 64640, 2048,
  3214, 1968, 6272, 64462, 6096, 1074, 8174, 63502, 2096, 1088, 7086, 64478, 3200, 144, 4302, 62622,
  2976, 63440, 4222, 60462, 64496, 64448, 6096, 63374, 34, 64464, 6082, 60590, 976, 63618, 5202, 58432,
  2096, 864, 63550, 4976, 5010, 992, 64414, 5040, 2032, 1038, 64382, 4014, 3152, 94, 61598, 4192,
  3954, 64430, 62526, 4064, 65458, 910, 65438, 4928, 2020, 926, 65406, 2128, 1938, 48, 64512, 992,
```

2032, 1874, 8176, 65360, 3952, 1968, 6064, 63440, 1982, 1040, 7054, 64382, 1086, 110, 5246, 63550,
65470, 64416, 4158, 61438, 64418, 65408, 5056, 61296, 962, 65408, 4992, 60496, 850, 64514, 6114, 62304,
32, 65392, 63534, 2928, 1954, 65504, 62446, 2048, 992, 78, 63422, 4016, 112, 158, 60606, 3184,
65522, 63470, 60526, 1056, 64482, 65456, 62462, 3984, 2020, 65440, 63392, 1152, 2948, 32, 63488, 2946,
1088, 1936, 5248, 64432, 3054, 1058, 7152, 62558, 64622, 1138, 5150, 62510, 2208, 1250, 3390, 62718,
46, 62496, 4256, 60528, 64514, 64512, 4112, 60416, 63570, 64498, 5090, 61536, 946, 64610, 5170, 61376,
65506, 62256, 59438, 1872, 65408, 64432, 61358, 1008, 62464, 64542, 61374, 1984, 82, 63630, 58604, 2208,
65474, 62430, 59486, 32, 65458, 63392, 62398, 1936, 64500, 65408, 62334, 2048, 1860, 65520, 62416, 1874,
4082, 1858, 6144, 64352, 4992, 2016, 6080, 64448, 2048, 1072, 5072, 63470, 2190, 1168, 4270, 62638,
2032, 62464, 3200, 60464, 65490, 64466, 3072, 62350, 64480, 65408, 2944, 61440, 64370, 63490, 4066, 60288,
4098, 832, 63534, 3952, 5008, 2016, 64478, 4048, 3090, 1088, 64494, 5118, 4258, 2208, 63694, 4272,
2064, 64528, 62638, 3136, 3058, 2016, 62494, 6032, 3060, 2960, 65424, 5136, 2946, 1040, 65520, 3984,
992, 1858, 4096, 63296, 64414, 978, 5008, 62448, 63504, 64592, 4048, 62432, 64670, 160, 2256, 61614,
65504, 63472, 2144, 59440, 62432, 64416, 4018, 61328, 64466, 65394, 2930, 61440, 64322, 63490, 3042, 59248,
63472, 63280, 60430, 1856, 64400, 65488, 62366, 992, 63488, 64592, 61406, 3024, 64656, 144, 59614, 3232,
992, 64480, 60526, 1056, 64482, 912, 63422, 3968, 2004, 1904, 62334, 6130, 2884, 2, 63456, 2930,
62526, 1922, 1120, 63376, 64464, 18, 3040, 61456, 63518, 64624, 4048, 62430, 62672, 64704, 1232, 60638,
62464, 61456, 2128, 59470, 63440, 64466, 3040, 61328, 61490, 64466, 1970, 59472, 64368, 63554, 4066, 60336,
32, 65408, 60510, 3968, 1986, 2, 62414, 3072, 2032, 1104, 63422, 6048, 176, 2222, 61630, 5296,
3042, 1008, 63536, 3104, 2994, 2992, 65486, 7024, 2066, 2992, 65454, 6208, 5988, 4128, 2016, 6050,
4176, 5026, 7234, 1952, 4032, 3090, 6064, 64496, 4048, 4098, 5022, 65456, 3136, 3200, 3216, 63648,
2976, 1936, 5120, 62464, 65504, 1986, 6080, 65408, 65522, 1984, 2992, 63584, 64432, 1122, 3122, 62448,
64578, 64384, 60462, 1920, 928, 63488, 60318, 2000, 1970, 2000, 61294, 2944, 2082, 80, 59502, 2160,
2930, 864, 63440, 4032, 64450, 928, 63376, 4962, 2004, 912, 63376, 4160, 1956, 1088, 63520, 3024,
62688, 3106, 1264, 64512, 2, 2130, 5104, 62528, 63520, 1152, 4064, 65440, 1104, 1200, 2224, 63632,

```
  65472, 898, 2128, 61456, 64450, 1968, 2064, 63408, 62496, 2034, 1056, 62624, 62496,
194, 3234, 60528,
  61632, 63488, 57566, 4066, 994, 64560, 61390, 2048, 65506, 64, 61358, 7010, 3088, 1120,
60542, 5202,
  1922, 2880, 61470, 6096, 1938, 3952, 63470, 8064, 3058, 4032, 62464, 7296, 4082, 4256,
128, 6224,
  61632, 6130, 3248, 2016, 65504, 4130, 6032, 992, 65456, 6082, 5936, 848, 1008, 5088,
6112, 2,
  1824, 3858, 6048, 864, 65346, 3906, 6016, 816, 946, 5010, 4048, 96, 992, 4242, 4224,
64528,
  59648, 63568, 57630, 2098, 63538, 64624, 61422, 1072, 64496, 2048, 62334, 6032, 48,
2080, 62494, 6192,
  2898, 2880, 64478, 8096, 1922, 4976, 64462, 9090, 3076, 6096, 63534, 9394, 4162, 6386,
192, 8210,
  63664, 5220, 1360, 64640, 64624, 2226, 1056, 64560, 62464, 3104, 1920, 848, 1008, 6114,
1054, 2016,
  1840, 3858, 4080, 65440, 880, 4994, 4080, 65440, 62528, 3090, 1104, 288, 62608, 5426,
64640, 912,
  59410, 63426, 53536, 64594, 60482, 61568, 54272, 1008, 62416, 65520, 58190, 2832,
1968, 2976, 60382, 6048,
  2802, 2768, 61358, 5984, 1858, 2880, 62400, 7026, 1042, 4064, 61472, 4320, 2146, 5328,
64464, 7904
  //date ends here
};
//**********************end of factort calibration
data**********************************************************************************************
void(* resetFunc) (void) = 0; //declare reset function @ address 0 //we reset if needed for hz
change, allows reboot
void setup()
{
  pinMode(A5, INPUT);          // set pin to input
  digitalWrite(A5, HIGH);      // turn on pullup resistors
  pinMode(A4, INPUT);          // set pin to input
  digitalWrite(A4, HIGH);      // turn on pullup resistor




  //while (!Serial); //Wait for user to open terminal
  // MLX90640_I2CWrite(slaveAddr, 0x8000, 0x0030);


  Wire.begin(MLX90640_address);
  Wire.setClock(400000); //Increase I2C clock speed to 400kHz
  Serial.begin(115200);//we init this first because i think it gives priority on data bus
  for (byte i = 0; i < 64; i++) {
    Serial.println(); //we clear screen in terminal in case reset.
  }
  //Serial.println(F("MLX90640 IR Array Example"));
  //  if (isConnected() == false)
```

```
  // {
  //   Serial.println("MLX90640 not detected at default I2C address. Please check wiring.
Freezing.");
  //   while (1);
  // }

  //Serial.println("MLX90640 online!");
  delay(500);//make sure serial done
  Wire.onReceive(receiveEvent); // register event
  // uint16_t eeMLX90640[832];
  //status = MLX90640_DumpEE(MLX90640_address, eeMLX90640);
  if (status != 0)
    Serial.println(F("Failed to load system parameters"));

  //  status = MLX90640_ExtractParameters(eeMLX90640, &mlx90640);
  if (status != 0)
    Serial.println(F("Parameter extraction failed"));

  //here we read eeprom
  //Serial.println();
  //Serial.println(F("This shows a single image from entire sensor. since we have the start
count delay, the memory will already be filled with sensor data"));
  //Serial.println();
  delay(200);//serial data to clear
}
//long  timert =millis();
uint16_t mydata[32];//array
uint16_t startvalue[32];//array
uint16_t worddata[1];
float ta ;//we use 4 corner sensors to determin chip temp at start up, then we take lowest pixel
on screen and use that.
float tr;//reflecitve temp. this is open air shift. going with documentation and using -8
float vddpage0;//we need voltage per page
float vddpage1;//we need voltage per page
float emissivity = 0.95;
bool runonce = true; //singleshot
void loop()
{ if (runonce) {
    for (int i = 0; i < 832; i++) {
      MLX90640_I2CRead(MLX90640_address, 0x2400 + i, 1, worddata);
      delay(1);
      //Serial.print(worddata[0]);
      //Serial.print((", "));delay(1);
      if ((i & 15) == 15) {
        //Serial.print(F(" register
location:"));Serial.print(i+0x2400-15,HEX);Serial.print("-");Serial.println(i+0x2400,HEX);
      }//every 8 data make a line
      delay(2);
    }//next
    //Serial.println(F("data above is from epprom. it will be used later on for calibration"));
```

```
   //Serial.println(F("We will now test the eeprom of the sensor with the flash data. if it does not match it will error but keep working using the included settings."));

   //    for (int i=0;i<832;i++){
   //      MLX90640_I2CRead(MLX90640_address, 0x2400+i, 1,worddata);
   //      if (worddata[0] !=pgm_read_word_near(factoryCalData+i)){Serial.print(F("Error at MemoLocation (in HEXMODE):"));
   //  Serial.print(0x2400+i,HEX);Serial.print("sensor:");Serial.print(worddata[0],HEX);
   //  Serial.print("PROGMEMVALUE:");Serial.println(pgm_read_word_near(factoryCalData+i),HEX);delay(2);//allows serial to finish before i2c read
   //      }
   //    }//next
   //    Serial.print(F("Flash memory check complete. if errors troubleshoot and fix for best results"));

   byte  testMode = MLX90640_GetCurMode(MLX90640_address);
   //Serial.print(F("Device is in "));
   //if (testMode==0){Serial.print(F("interleaved"));}
   //if (testMode==1){Serial.print(F("checkered  pattern"));}
   //Serial.println(F("mode"));
   testMode = MLX90640_GetCurResolution(MLX90640_address); //we get resolution of ad converter 16-19 bit resolution
   //Serial.print(F("Device ad resoltuion set to ...."));
   //if (testMode==0){Serial.print(F("16"));}
   //if (testMode==1){Serial.print(F("17"));}
   //if (testMode==2){Serial.print(F("18"));}
   //if (testMode==3){Serial.print(F("19"));}
   //Serial.println(F(".... bit"));
   //    if (adSensorResolution==testMode){Serial.println(F("analog resolution is set the same as #define settings in flash"));}
   //    else{//if resolution is different we change resolution
   //      MLX90640_I2CRead(MLX90640_address,  0x800D,  1,worddata);//we read control register
   //      delay(1);
   //      worddata[0]=worddata[0]&62463;//1111 0011 1111 1111 //we clear bits so we can just add changes
   //      switch (adSensorResolution){
   //      case 0: break; //we do nothing on zero 16bit
   //      case 1:worddata[0]=worddata[0]+1024; break; //we set bit 17bit
   //      case 2:worddata[0]=worddata[0]+2048; break; //we set bit 18bit
   //      case 3:worddata[0]=worddata[0]+2048+1024; break; //we set bits 19bit
   //      }
   //      MLX90640_I2CWrite(MLX90640_address, 0x800D, worddata[0]);//we write modified data back to register and make it single mode
   //      delay(5);// we add delay soit can finish write
   //      Serial.println(F("ad resolution now changed to match flash"));
   //      delay(200);
   //      Serial.println(F("we are resetting to make sure changes stick"));
   //      delay(2000);
```

```
    //      resetFunc();  //call reset
    //    }

    testMode = MLX90640_GetRefreshRate(MLX90640_address); //get refresh in hz
    //    Serial.print(F("HZ is set to "));
    //    if (testMode==0){Serial.print(F("0.5"));}
    //    if (testMode==1){Serial.print(F("1.0"));}
    //    if (testMode==2){Serial.print(F("2.0"));}
    //    if (testMode==3){Serial.print(F("4.0"));}
    //    if (testMode==4){Serial.print(F("8.0"));}
    //    if (testMode==5){Serial.print(F("16.0"));}
    //    if (testMode==6){Serial.print(F("32.0"));}
    //    if (testMode==7){Serial.print(F("64.0"));}
    //    Serial.print(F("HZ"));
    if (hzMode==testMode){;} // {Serial.println(F("HZ is set the same as #define settings in
flash"))
    else{//this means setting is different
    //Serial.println(F("HZ setting is different in firmware. we are going to write data now."));

    MLX90640_I2CRead(MLX90640_address,  0x800D,  1, worddata); //we read control
register
    delay(1);
    worddata[0] = worddata[0] & 64639 ; //we make 1111110001111111 the zeros are the control
registers for refresh rate. we set zero here for ease of or of 1's later
    switch (hzMode) {
      case 0: break;//we dont change anything
      case 1: worddata[0] = worddata[0] + 128;   break;
      case 2: worddata[0] = worddata[0] + 256;   break;
      case 3: worddata[0] = worddata[0] + 256 + 128;   break;
      case 4: worddata[0] = worddata[0] + 512;   break;
      case 5: worddata[0] = worddata[0] + 512 + 128;   break;
      case 6: worddata[0] = worddata[0] + 512 + 256;   break;
      case 7: worddata[0] = worddata[0] + 512 + 256 + 128;   break;
        //we can simplify above later. this is not much of a concern right now. this seems fast for
a single change test
    }
    //ok we have data changes to matcht the new time change, now we need to write it
    MLX90640_I2CWrite(MLX90640_address, 0x800D, worddata[0]);//we write modified data
back to register and make it single mode
    delay(5);// we add delay soit can finish write
    Serial.println(F("HZ now changed to match flash"));
    delay(200);
    //Serial.println(F("rebooting so we can verify it took."));
    delay(2000);
    resetFunc();  //call reset
  }//end of else
  delay(200);
  //Serial.print(F("register:"));
  delay(10);
  MLX90640_I2CRead(MLX90640_address,  0x8000,  1, worddata);
```

```
  delay(10);
  MLX90640_I2CRead(MLX90640_address,  0x800D,  1, worddata); //we get status of step mode. default it is not in step mode, but we only want to write to eeprom if it is not in step mode
  delay(5);//we wait ample time for i2c to complete
  printBits(highByte(worddata[0]));
  printBits(lowByte(worddata[0]));
  if ((worddata[0] & 2) == 0) {
    //Serial.println(F(" Device is in continueous mode"));
    if (!continuousmode) { //here is where we change it if it is set different
      Serial.println(F("We need to change mode to step mode as per #define continueousmode in sketch"));
      delay(5);//we wait ample time for i2c to complete

      worddata[0] = worddata[0] | 2; //we make this and 0000000000000010; bit2 is now high this is needed at 800D HEX location
      delay(5);
      Serial.println("new values:");
      printBits(highByte(worddata[0]));
      printBits(lowByte(worddata[0]));
      Serial.println();
      delay(5000);
      MLX90640_I2CWrite(MLX90640_address, 0x800D, worddata[0]);//we write modified data back to register and make it single mode
      delay(5);//we wait ample time for i2c to complete
      Serial.println(F("Data should now be changed."));
      delay(5);//we wait ample time for i2c to complete
    }
  }

  else
  { Serial.println(F(" Device is in step mode"));
    if (continuousmode) { //here is where we change it if it is set different
      Serial.println(F("We need to change mode to  continueous mode as per #define continueousmode in sketch"));
      delay(5);//we wait ample time for i2c to complete
      worddata[0] = worddata[0] & 65533; //we make this and 1111111111111101; bit2 is now low this is needed at 800D HEX location
      delay(5);
      Serial.println("new values:");
      printBits(highByte(worddata[0]));
      printBits(lowByte(worddata[0]));
      Serial.println();
      delay(5000);
      MLX90640_I2CWrite(MLX90640_address, 0x800D, worddata[0]);//we write modified data back to register and make it single mode
      delay(5);//we wait ample time for i2c to complete
      Serial.println(F("Data should now be changed."));
      delay(5);//we wait ample time for i2c to complete
    }
```

```
  }
  delay(10);
  //this prints out register data
  printBits(highByte(worddata[0]));
  printBits(lowByte(worddata[0]));
  //Serial.print(F(". This means that page is on:"));
  //Serial.println(1 & worddata[0]); //it can be zero or 1
//  Serial.print(F("seconds before startup:7..")); delay(1000);
//  Serial.print(F("6.")); delay(1000);
//  Serial.print(F("5.")); delay(1000);
//  Serial.print(F("4.")); delay(1000);
//  Serial.print(F("3.")); delay(1000);
//  Serial.print(F("2.")); delay(1000);
//  Serial.println(F("1.")); delay(1000);
  //we force two scans so we get preuse data
  if (!continuousmode) { //if we are in step mode we need to tell sensor to scan page
    MLX90640_I2CRead(MLX90640_address,  0x8000,  1, worddata); //we read sensor
    delay(600);
    worddata[0] =  worddata[0] | 32; //we set 000000000100000 wich is register that starts
measurement
    MLX90640_I2CWrite(MLX90640_address, 0x8000, worddata[0]);//we write modified values
    delay(600);
    worddata[0] =  worddata[0] | 32; //we set 000000000100000 wich is register that starts
measurement
    MLX90640_I2CWrite(MLX90640_address, 0x8000, worddata[0]);//we write modified values
  }

  MLX90640_I2CRead(MLX90640_address,  0x0400,  32, startvalue);
  runonce = false;
}//we get values one time only

for (int i = 0; i < 24; i++) { //this samples and draws entire page of data. sensor is
  MLX90640_I2CRead(MLX90640_address,  0x0400 + 32 * i,  32, mydata); //read 10 places in
memory
  for (int x = 0 ; x < 32 ; x += 1) {
    //we compareative one row to see at output
    int testx = (50 + mydata[x] - startvalue[x]);
    char outputx = 46;
    //if (testx>60){outputx=44;}//+
    //if (testx>70){outputx=111;}//o
    //if (testx>80){outputx=79;}//O
    //if (testx>100){outputx=64;}//@
    //Serial.print(x); Serial.print(" "); Serial.print(i); Serial.print(" "); Serial.print(testx);
Serial.println();
    Serial.print(testx); Serial.print(" ");

    //Serial.print  (outputx);Serial.print  ("  "); //old output
  }
  Serial.println();
  // for a line between every row
```

```
  //MLX90640_I2CRead(MLX90640_address,  0x8000,  1,worddata);
  //delay(50);
  //Serial.print(F("mempage (we read all 32 even if old data):"));
  //Serial.println(1&worddata[0]);//it can be zero or 1
}// do loop thru all 24 lines!
  //delay(100);//allow serial to finish
  //Serial.println();

// REMOVE double space between outputs
//Serial.print(0);


//we alread see screen. now lets refresh it before we look at data again
if (!continuousmode) { //if we are in step mode we need to tell sensor to scan page
  MLX90640_I2CRead(MLX90640_address,  0x8000,  1, worddata); //we read sensor

  worddata[0] =  worddata[0] | 32; //we set 000000000100000 wich is register that starts
measurement
  worddata[0] =  worddata[0] | 16; //we set 000000000101000 auto next page
  delay(5);
  MLX90640_I2CWrite(MLX90640_address, 0x8000, worddata[0]);//we write modified values
  delay(600);
  worddata[0] =  worddata[0] | 32; //we set 000000000100000 wich is register that starts
measurement
  worddata[0] =  worddata[0] | 16; //we set 000000000101000 auto next page
  MLX90640_I2CWrite(MLX90640_address, 0x8000, worddata[0]);//we write modified values
  delay(600);
}
//Serial.println(F("pages done. if in step mode memory page will always read 1, because it
scans twice and ends up on page 1"));
delay(1500);//2hz is speed of sensor read by defaul
}
```

GNUPLOT script

This script is run using the Cygwin64 Terminal under a windows environment. This is needed because the script uses the UNIX tool *head* and *tail*. This script excepts that the sensor data is being written into a log file. Currently PuTTY is used to log data from the COM port but this is less than ideal: it leads to deadlock if we try and read this file when the data is being written. If the frequency of data write from the board to the computer via the COM port is out of sync with the refresh rate of the GNUplot script, you can get a scroll effect like an old TV. This issue is solely with the way data is fed to the script.

```
# Stephen Duncanson
# HOME BODY UConn Senior Design 2021-2022
# Plot sensor values using gnuplot for testing
# this assumes that data is stored in columns in a file called "source.dat"
# Aside: how we get "source.dat" with arduino mega2560 connected to computer via serial
comm port
# Answer: Using PuTTY
# Session: connection type: Serial, set speed and line
# logging
# the first two columns contain independent variables (row, col) index into the IR array
# the third column contains the sensor value for that pixel.

# BOTH VERSIONS
unset border
unset tics
#unset key
set cbrange [-1:100]            # for just below freezing -1C to 100C
set xrange [0:31]
set yrange [0:23]

set view map
#set pm3d map
#set pm3d interpolate 3,3

#system "(tail -793 source.txt | head -792) > source2.txt"
system "(tail -25 source.txt | head -24) > source2.txt"

plot "source2.txt" matrix using 1:2:3 with image

# change so its while we are ready to get data?
while (1) {
    pause 1
    system "(tail -25 source.txt | head -24) > source2.txt"
    replot
}
```