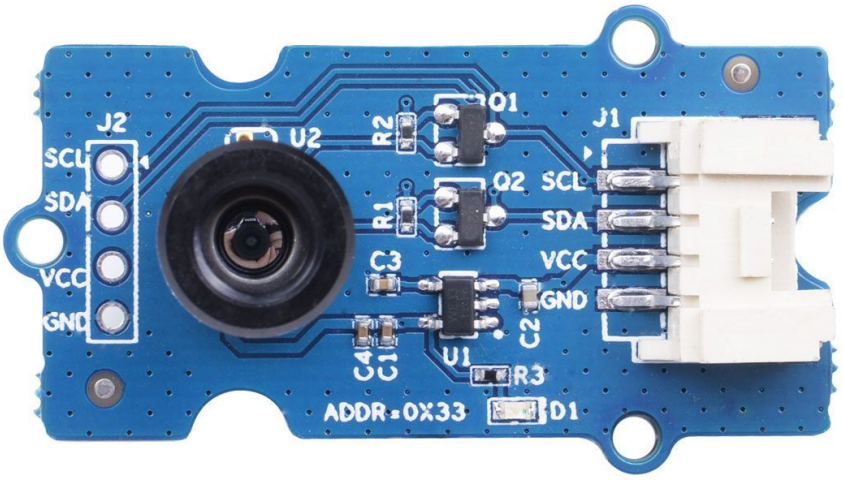
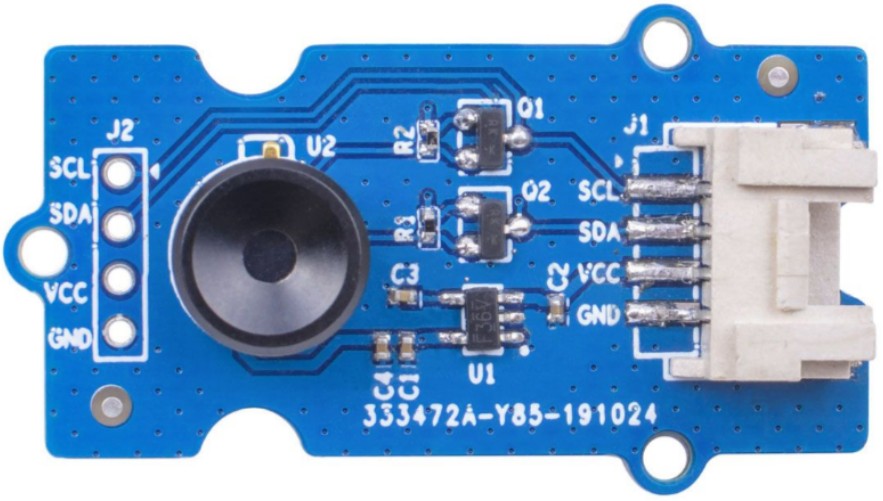
Grove - Thermal Imaging Camera IR-Array MLX90641



Grove - Thermal Imaging Camera / IR Array MLX90641

This IR thermal camera carries a 16x12 array of thermal sensors **(MLX90641)** and it can detect the temperature of objects from far away with a center area accuracy of ±1℃ and average accuracy of ±1.5℃. In order to obtain the thermal images easily, the I2C

protocol is used to get the low-resolution images from the camera. The FOV (Field of View) of this camera is 110°x75°, and the temperature measurement range is -40℃ to 300℃. In order to obtain the thermal image easily, I2C protocol is used to get the low-resolution image from the camera.



Grove - Thermal Imaging Camera / IR Array MLX90640

While Grove - Thermal Imaging Camera is a thermal sensor **(MLX90640)** , carrying a 32x24 array of thermal sensors, and it can detect the temperature of objects from feet away with the accuracy of ±1.5℃ and can present dynamic thermal images and detect the surrounding temperature from -40℃~300℃. The camera with narrow-angle/wide-angle has an FOV(Field of View) of 55°x35°/110°x75°.In order to obtain the thermal image easily, I2C protocol is used to get the low-resolution image from the camera.

Versions

Version	Date of Released	Order
Grove - Thermal Imaging Camera / IR Array **MLX90641** 110 degree ***[New]***	03-June-2020	Buy it [https://www.seeedstudio.com/Grove-Thermal-Imaging-Camera-IR-Array-MLX90641-110-degree-p-4612.html]
Grove - Thermal Imaging Camera / IR Array **MLX90640** 110 degree	12-Nov-2019	Buy it [https://www.seeedstudio.com/Grove-Thermal-Imaging-Camera-IR-Array-MLX90640-110-degree-p-4334.html]

Note
This wiki fits both types of the Thermal Imageing Camera IR Array MLX90641 and MLX90640.

Features

- Compact size 16x12 pixel IR thermal sensor array (MLX90641)，32x24 array pixel IR thermal sensor array (MLX90640)
- High FOV (field-of-view) of 110°x75° to capture more area
- Wide temperature measurement range (-40℃~300℃)
- I2C Grove interface for easy communication with an MCU
- Fully calibrated IR array for convenient setup

Specification

Item	Grove - Thermal Imaging Camera - MLX90640	Grove - Thermal Imaging Camera - MLX90641
Thermal sensor	32X24 array MLX90640	16x12 array MLX90641
Operating Voltage	3.3V - 5V	3.3V - 5V
Current consumption	~18mA	~18mA
FOV(Field of View)	110°x75°	110°x75°
Temperature Measurement Range	-40°C - 300°C	-40°C - 300°C
Temperature Resolution	± 1.5°C	± 1.5°C (±1℃ at center area)
Refresh Rate	0.5Hz - 64Hz	0.5Hz - 64Hz
Interface	I2C Grove interface	I2C Grove interface
I2C Address	0x33	0x33

Platforms Supported

Arduino	Raspberry Pi		

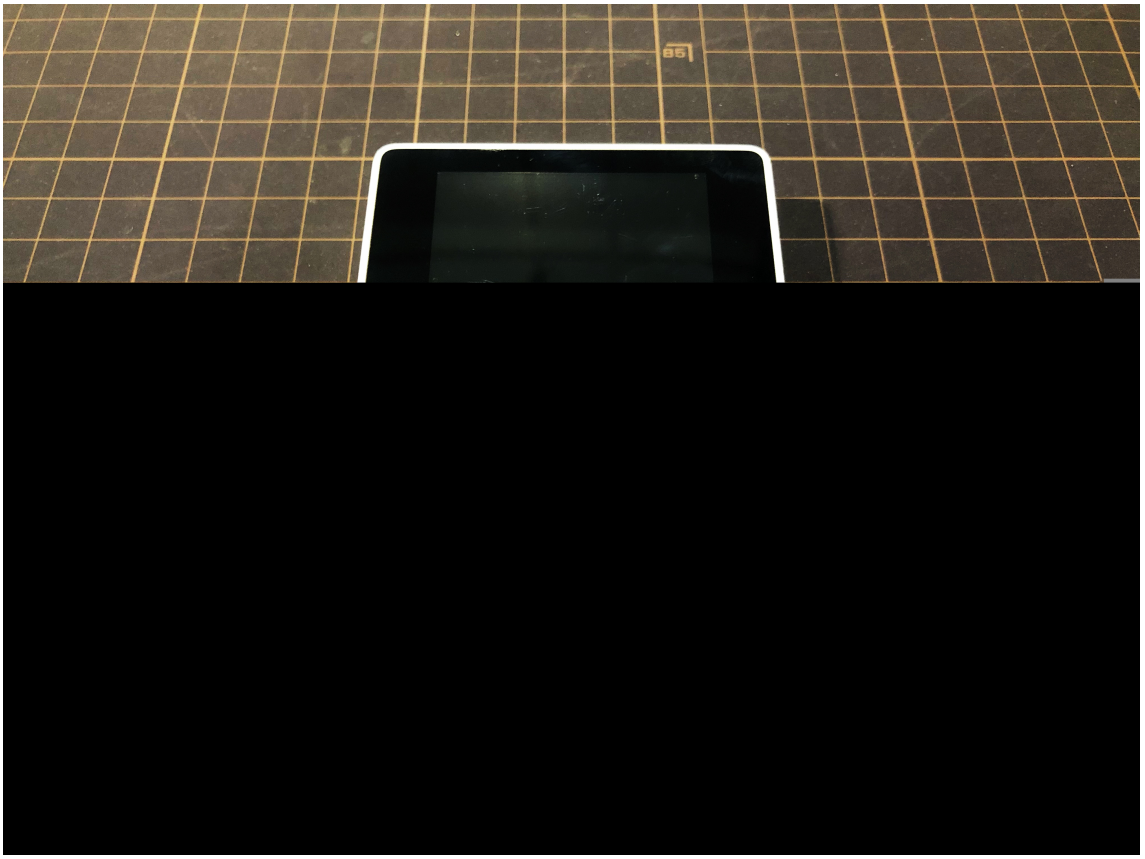
Getting Started

Getting Started by Wio Terminal

Materials required

Wio Terminal	Grove - Thermal Imaging Camera / IR Array MLX90641 110 degree
	
Get one now [https://www.seeedstudio.com/Wio-Terminal-p-4509.html]	Get one now [https://www.seeedstudio.com/Grove-Thermal-Imaging-Camera-IR-Array-MLX90641-110-degree-p-4612.html]

Hardware Connection



Step 1. Plug Grove - Thermal Imaging Camera to Wio Terminal via Grove cable and also connect Wio Terminal to PC through a USB cable.

Step 2. Download the Library [https://github.com/Seeed-Studio/Seeed_Arduino_MLX9064x/archive/master.zip] and copy the whole **Seeed_Arduino_MLX9064x** file and paste it into your Arduino IDE library file.

Note
If it is your first time playing Wio Terminal and not sure which interface to plug on Wio Terminal, please refer to **Get Started with Wio Terminal [https://wiki.seeedstudio.com/Wio-Terminal-Getting-Started/]**.

Step 3. Copy the Software Code 1 below into your Arduino IDE and upload it for visualization format displayed via **Serial Port.**

Outcome of Visualization Format

Software Code 1

```
/*
    Output the temperature readings to all pixels to be
*/

#include <Wire.h>

#define USE_MLX90641

#ifndef USE_MLX90641
    #include "MLX90640_API.h"
#else
    #include "MLX90641_API.h"
#endif

#include "MLX9064X_I2C_Driver.h"

#if defined(ARDUINO_ARCH_AVR)
    #define debug  Serial

#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_AR
    #define debug  Serial
#else
    #define debug  Serial
#endif

#ifdef USE_MLX90641
    const byte MLX90641_address = 0x33; //Default 7-bit
    #define TA_SHIFT 8 //Default shift for MLX90641 in

    uint16_t eeMLX90641[832];
    float MLX90641To[192];
    uint16_t MLX90641Frame[242];
```

```
    paramsMLX90641 MLX90641;
    int errorno = 0;
#else
    const byte MLX90640_address = 0x33; //Default 7-bit

    #define TA_SHIFT 8 //Default shift for MLX90640 in

    float mlx90640To[768];
    paramsMLX90640 mlx90640;
#endif
void setup() {
    Wire.begin();
    Wire.setClock(400000); //Increase I2C clock speed t

    debug.begin(115200); //Fast debug as possible

    while (!debug); //Wait for user to open terminal
    //debug.println("MLX90640 IR Array Example");


#ifndef USE_MLX90641
    if (isConnected() == false) {
        debug.println("MLX9064x not detected at default
        while (1);
    }
    //Get device parameters - We only have to do this o
    int status;
    uint16_t eeMLX90640[832];
    status = MLX90640_DumpEE(MLX90640_address, eeMLX906
    if (status != 0) {
        debug.println("Failed to load system parameters
    }

    status = MLX90640_ExtractParameters(eeMLX90640, &ml
    if (status != 0) {
        debug.println("Parameter extraction failed");
    }

    //Once params are extracted, we can release eeMLX90

    //MLX90640_SetRefreshRate(MLX90640_address, 0x02);
```

```
74      MLX90640_SetRefreshRate(MLX90640_address, 0x03); //
75      //MLX90640_SetRefreshRate(MLX90640_address, 0x07);
76  #else
77      if (isConnected() == false) {
78          debug.println("MLX90641 not detected at default
79          while (1);
80      }
81      //Get device parameters - We only have to do this o
82      int status;
83      status = MLX90641_DumpEE(MLX90641_address, eeMLX906
84      errorno = status;//MLX90641_CheckEEPROMValid(eeMLX9
85
86      if (status != 0) {
87          debug.println("Failed to load system parameters
88         while(1);
89      }
90
91      status = MLX90641_ExtractParameters(eeMLX90641, &ML
92      //errorno = status;
93      if (status != 0) {
94          debug.println("Parameter extraction failed");
95          while(1);
96      }
97
98      //Once params are extracted, we can release eeMLX90
99
100     //MLX90641_SetRefreshRate(MLX90641_address, 0x02);
101     MLX90641_SetRefreshRate(MLX90641_address, 0x03); //
102     //MLX90641_SetRefreshRate(MLX90641_address, 0x07);
103 #endif
104
105 }
106
107 void loop() {
108 #ifndef USE_MLX90641
109     long startTime = millis();
110     for (byte x = 0 ; x < 2 ; x++) {
111         uint16_t mlx90640Frame[834];
112         int status = MLX90640_GetFrameData(MLX90640_add
113
114         float vdd = MLX90640_GetVdd(mlx90640Frame, &mlx
```

```
115         float Ta = MLX90640_GetTa(mlx90640Frame, &mlx90
116
117         float tr = Ta - TA_SHIFT; //Reflected temperatu
118         float emissivity = 0.95;
119
120         MLX90640_CalculateTo(mlx90640Frame, &mlx90640,
121     }
122     long stopTime = millis();
123
124     for (int x = 0 ; x < 768 ; x++) {
125         //if(x % 8 == 0) debug.println();
126         debug.print(mlx90640To[x], 2);
127         debug.print(",");
128     }
129     debug.println("");
130 #else
131     long startTime = millis();
132
133     for (byte x = 0 ; x < 2 ; x++) {
134         int status = MLX90641_GetFrameData(MLX90641_add
135
136         float vdd = MLX90641_GetVdd(MLX90641Frame, &MLX
137         float Ta = MLX90641_GetTa(MLX90641Frame, &MLX90
138
139         float tr = Ta - TA_SHIFT; //Reflected temperatu
140         float emissivity = 0.95;
141
142         MLX90641_CalculateTo(MLX90641Frame, &MLX90641,
143     }
144     long stopTime = millis();
145    /*
146     debug.print("vdd=");
147     debug.print(vdd,2);
148     debug.print(",Ta=");
149     debug.print(Ta,2);
150
151     debug.print(",errorno=");
152     debug.print(errorno,DEC);
153
154
155     for (int x = 0 ; x < 64 ; x++) {
```

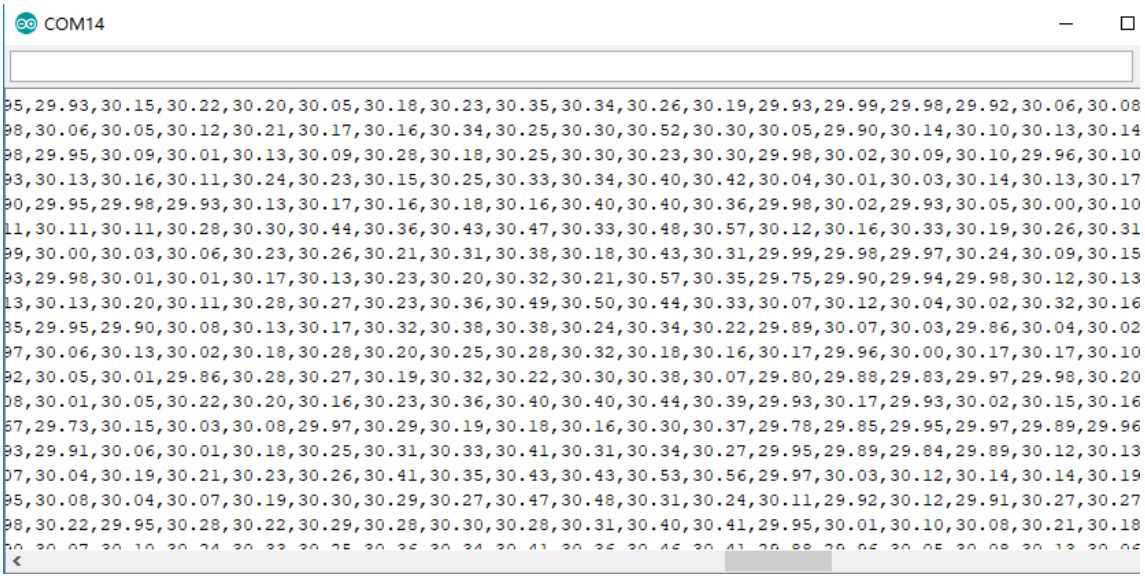
```
            debug.print(MLX90641Frame[x], HEX);
            debug.print(",");
        }

        delay(1000);
        */
        for (int x = 0 ; x < 192 ; x++) {
            debug.print(MLX90641To[x], 2);
            debug.print(",");
        }
        debug.println("");
    #endif
    }

    //Returns true if the MLX90640 is detected on the I2C b
    boolean isConnected() {
    #ifndef USE_MLX90641
        Wire.beginTransmission((uint8_t)MLX90640_address);
    #else
        Wire.beginTransmission((uint8_t)MLX90641_address);
    #endif
        if (Wire.endTransmission() != 0) {
            return (false);    //Sensor did not ACK
        }
        return (true);
    }
```

Note

Upload the software code 1 above into your Arduino IDE and open the **Serial Port**, you will see an outcome of visualization format as following:



Outcome of Visualization on Wio Terminal

Step 4. Upload the Software Code 2 below into your Arduino IDE for visualization displayed on Wio Terminal.

Software Code 2

```
#include <Wire.h>
#include "MLX90641_API.h"
#include "MLX9064X_I2C_Driver.h"
#include <TFT_eSPI.h>                    // Include the gra

const byte MLX90641_address = 0x33; //Default 7-bit uns
#define TA_SHIFT 12 //Default shift for MLX90641 in ope
#define debug  Serial
uint16_t eeMLX90641[832];
float MLX90641To[192];
uint16_t MLX90641Frame[242];
paramsMLX90641 MLX90641;
int errorno = 0;

TFT_eSPI    tft = TFT_eSPI();
TFT_eSprite Display = TFT_eSprite(&tft);  // Create Spr
// the pointer is used by pushSprite() to push it onto

```

```
unsigned long CurTime;

uint16_t TheColor;
// start with some initial colors
uint16_t MinTemp = 25;
uint16_t MaxTemp = 38;

// variables for interpolated colors
byte red, green, blue;

// variables for row/column interpolation
byte i, j, k, row, col, incr;
float intPoint, val, a, b, c, d, ii;
byte aLow, aHigh;

// size of a display "pixel"
byte BoxWidth = 3;
byte BoxHeight = 3;

int x, y;
char buf[20];

// variable to toggle the display grid
int ShowGrid = -1;

// array for the interpolated array
float HDTemp[6400];

void setup() {
    Wire.begin();
    Wire.setClock(2000000); //Increase I2C clock speed
    debug.begin(115200); //Fast debug as possible

    // start the display and set the background to blac

    if (isConnected() == false) {
        debug.println("MLX90641 not detected at default
        while (1);
    }
    //Get device parameters - We only have to do this o
    int status;
```

```
60      status = MLX90641_DumpEE(MLX90641_address, eeMLX906
61      errorno = status;//MLX90641_CheckEEPROMValid(eeMLX9
62
63      if (status != 0) {
64          debug.println("Failed to load system parameters
65         while(1);
66      }
67
68      status = MLX90641_ExtractParameters(eeMLX90641, &ML
69      //errorno = status;
70      if (status != 0) {
71          debug.println("Parameter extraction failed");
72          while(1);
73      }
74
75      //Once params are extracted, we can release eeMLX90
76
77      MLX90641_SetRefreshRate(MLX90641_address, 0x05); //
78
79      tft.begin();
80      tft.setRotation(3);
81      tft.fillScreen(TFT_BLACK);
82      Display.createSprite(TFT_HEIGHT, TFT_WIDTH);
83      Display.fillSprite(TFT_BLACK);
84
85      // get the cutoff points for the color interpolatio
86      // note this function called when the temp scale is
87      Getabcd();
88
89      // draw a legend with the scale that matches the se
90      DrawLegend();
91  }
92  void loop() {
93      // draw a large white border for the temperature ar
94      Display.fillRect(10, 10, 220, 220, TFT_WHITE);
95      for (byte x = 0 ; x < 2 ; x++) {
96          int status = MLX90641_GetFrameData(MLX90641_add
97
98          float vdd = MLX90641_GetVdd(MLX90641Frame, &MLX
99          float Ta = MLX90641_GetTa(MLX90641Frame, &MLX90
100
```

```
            float tr = Ta - TA_SHIFT; //Reflected temperatu
            float emissivity = 0.95;

            MLX90641_CalculateTo(MLX90641Frame, &MLX90641,
        }

        interpolate_image(MLX90641To,12,16,HDTemp,80,80);

        //display the 80 x 80 array
        DisplayGradient();

        //Crosshair in the middle of the screen
        Display.drawCircle(115, 115, 5, TFT_WHITE);
        Display.drawFastVLine(115, 105, 20, TFT_WHITE);
        Display.drawFastHLine(105, 115, 20, TFT_WHITE);
        //Displaying the temp at the middle of the Screen

        //Push the Sprite to the screen
        Display.pushSprite(0, 0);

        tft.setRotation(3);
        tft.setTextColor(TFT_WHITE);
        tft.drawFloat(HDTemp[35 * 80 + 35], 2, 90, 20);

    }
    //Returns true if the MLX90640 is detected on the I2C b
    boolean isConnected() {
        Wire.beginTransmission((uint8_t)MLX90641_address);
        if (Wire.endTransmission() != 0) {
            return (false);    //Sensor did not ACK
        }
        return (true);
    }
    // function to display the results
    void DisplayGradient() {

      tft.setRotation(4);

      // rip through 70 rows
      for (row = 0; row < 70; row ++) {

```

```
142       // fast way to draw a non-flicker grid--just make e
143       // drawing lines after the grid will just flicker t
144       if (ShowGrid < 0) {
145         BoxWidth = 3;
146       }
147       else {
148         if ((row % 10 == 9) ) {
149           BoxWidth = 2;
150         }
151         else {
152           BoxWidth = 3;
153         }
154       }
155       // then rip through each 70 cols
156       for (col = 0; col < 70; col++) {
157
158         // fast way to draw a non-flicker grid--just make
159         if (ShowGrid < 0) {
160           BoxHeight = 3;
161         }
162         else {
163           if ( (col % 10 == 9)) {
164             BoxHeight = 2;
165           }
166           else {
167             BoxHeight = 3;
168           }
169         }
170         // finally we can draw each the 70 x 70 points, n
171         Display.fillRect((row * 3) + 15, (col * 3) + 15,
172       }
173     }
174
175   }
176   // my fast yet effective color interpolation routine
177   uint16_t GetColor(float val) {
178
179     /*
180       pass in value and figure out R G B
181       several published ways to do this I basically graph
182       again a 5-6-5 color display will not need accurate
```

```
183
184       equations based on
185       http://web-tech.ga-usa.com/2012/05/creating-a-custo
186
187     */
188
189     red = constrain(255.0 / (c - b) * val - ((b * 255.0)
190
191     if ((val > MinTemp) & (val < a)) {
192       green = constrain(255.0 / (a - MinTemp) * val - (25
193     }
194     else if ((val >= a) & (val <= c)) {
195       green = 255;
196     }
197     else if (val > c) {
198       green = constrain(255.0 / (c - d) * val - (d * 255.
199     }
200     else if ((val > d) | (val < a)) {
201       green = 0;
202     }
203
204     if (val <= b) {
205       blue = constrain(255.0 / (a - b) * val - (255.0 * b
206     }
207     else if ((val > b) & (val <= d)) {
208       blue = 0;
209     }
210     else if (val > d) {
211       blue = constrain(240.0 / (MaxTemp - d) * val - (d *
212     }
213
214     // use the displays color mapping function to get 5-6
215     return Display.color565(red, green, blue);
216
217   }
218
219   // function to get the cutoff points in the temp vs RGB
220   void Getabcd() {
221
222     a = MinTemp + (MaxTemp - MinTemp) * 0.2121;
223     b = MinTemp + (MaxTemp - MinTemp) * 0.3182;
```

```
  c = MinTemp + (MaxTemp - MinTemp) * 0.4242;
  d = MinTemp + (MaxTemp - MinTemp) * 0.8182;

}
float get_point(float *p, uint8_t rows, uint8_t cols, i
{
    if (x < 0)
    {
        x = 0;
    }
    if (y < 0)
    {
        y = 0;
    }
    if (x >= cols)
    {
        x = cols - 1;
    }
    if (y >= rows)
    {
        y = rows - 1;
    }
    return p[y * cols + x];
}

void set_point(float *p, uint8_t rows, uint8_t cols, in
{
    if ((x < 0) || (x >= cols))
    {
        return;
    }
    if ((y < 0) || (y >= rows))
    {
        return;
    }
    p[y * cols + x] = f;
}

// src is a grid src_rows * src_cols
// dest is a pre-allocated grid, dest_rows*dest_cols
void interpolate_image(float *src, uint8_t src_rows, ui
```

```
265                         float *dest, uint8_t dest_rows,
266 {
267     float mu_x = (src_cols - 1.0) / (dest_cols - 1.0);
268     float mu_y = (src_rows - 1.0) / (dest_rows - 1.0);
269 
270     float adj_2d[16]; // matrix for storing adjacents
271 
272     for (uint8_t y_idx = 0; y_idx < dest_rows; y_idx++)
273     {
274         for (uint8_t x_idx = 0; x_idx < dest_cols; x_id
275         {
276             float x = x_idx * mu_x;
277             float y = y_idx * mu_y;
278             get_adjacents_2d(src, adj_2d, src_rows, src
279 
280             float frac_x = x - (int)x; // we only need
281             float frac_y = y - (int)y; // we only need
282             float out = bicubicInterpolate(adj_2d, frac
283             set_point(dest, dest_rows, dest_cols, x_idx
284         }
285     }
286 }
287 
288 // p is a list of 4 points, 2 to the left, 2 to the rig
289 float cubicInterpolate(float p[], float x)
290 {
291     float r = p[1] + (0.5 * x * (p[2] - p[0] + x * (2.0
292     return r;
293 }
294 
295 // p is a 16-point 4x4 array of the 2 rows & columns le
296 float bicubicInterpolate(float p[], float x, float y)
297 {
298     float arr[4] = {0, 0, 0, 0};
299     arr[0] = cubicInterpolate(p + 0, x);
300     arr[1] = cubicInterpolate(p + 4, x);
301     arr[2] = cubicInterpolate(p + 8, x);
302     arr[3] = cubicInterpolate(p + 12, x);
303     return cubicInterpolate(arr, y);
304 }
305 
```

```
306 // src is rows*cols and dest is a 4-point array passed
307 void get_adjacents_1d(float *src, float *dest, uint8_t
308 {
309     // pick two items to the left
310     dest[0] = get_point(src, rows, cols, x - 1, y);
311     dest[1] = get_point(src, rows, cols, x, y);
312     // pick two items to the right
313     dest[2] = get_point(src, rows, cols, x + 1, y);
314     dest[3] = get_point(src, rows, cols, x + 2, y);
315 }
316
317 // src is rows*cols and dest is a 16-point array passed
318 void get_adjacents_2d(float *src, float *dest, uint8_t
319 {
320     float arr[4];
321     for (int8_t delta_y = -1; delta_y < 3; delta_y++)
322     {                                                  // -1, 0
323         float *row = dest + 4 * (delta_y + 1); // index
324         for (int8_t delta_x = -1; delta_x < 3; delta_x+
325         { // -1, 0, 1, 2
326             row[delta_x + 1] = get_point(src, rows, col
327         }
328     }
329 }
330
331 // function to draw a legend
332 void DrawLegend() {
333
334   //color legend with max and min text
335   j = 0;
336
337   float inc = (MaxTemp - MinTemp ) / 160.0;
338
339   for (ii = MinTemp; ii < MaxTemp; ii += inc) {
340     tft.drawFastHLine(260, 200 - j++, 30, GetColor(ii))
341   }
342
343   tft.setTextSize(2);
344   tft.setCursor(245, 20);
345   tft.setTextColor(TFT_WHITE, TFT_BLACK);
346   sprintf(buf, "%2d/%2d", MaxTemp, (int) (MaxTemp * 1.1
```

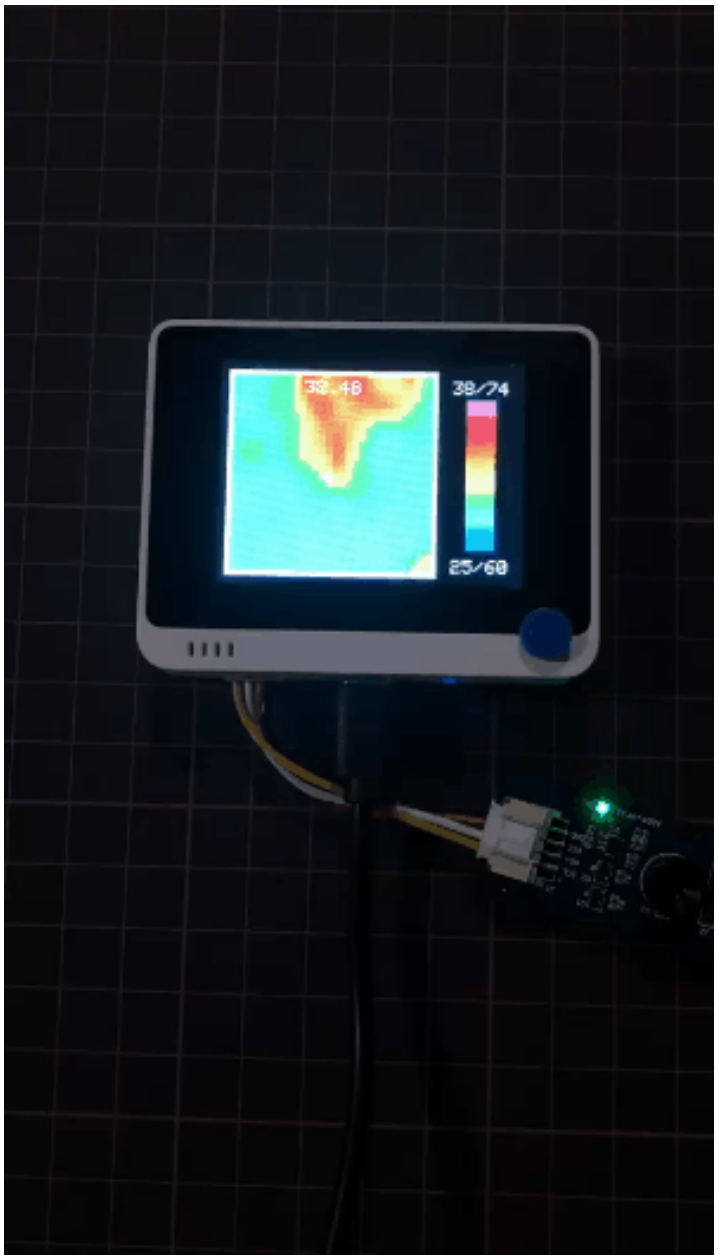
```
    tft.print(buf);

    tft.setTextSize(2);
    tft.setCursor(245, 210);
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
    sprintf(buf, "%2d/%2d", MinTemp, (int) (MinTemp * 1.1
    tft.print(buf);

}
```

Success

The outcome of visualization will display on the screen of Wio Terminal if everything goes well



Getting Started by Raspberry Pi

Hardware

Materials required

Raspberry Pi 4	Grove Base Hat for Raspberry Pi
	
Get one now [https://www.seeedstudio.com/Raspberry-Pi-4-Computer-Model-B-4GB-p-4077.html]	Get one now [https://www.seeedstudio.com/Grove Base-Hat-for-Raspberry-Pi.html]

Hardware Connection

- **Step 1** Connect the Grove - Thermal Imaging Camera to one of the two I2C ports.
- **Step 2** Plug the Raspberry Pi 4 into Grove Base Hat for Raspberry Pi.
- **Step 3** Connect the Raspberry Pi to a display via HDMI cable, and power on the Raspberry Pi 4 by USB type-C.

Software

Raspberry Pi 4 supports Python, so the project demo can be easily displayed from the Raspberry Pi 4 display if you follow the below steps.

- **Step 1** Install grove.py [https://github.com/Seeed-Studio/grove.py] by the command

```
pip3 install Seeed-grove.py
```

- **Step 2** Install the MLX90641 driver with the following command. Python environment(If you don't have authority of your Raspberry Pi):

```
pip3 install seeed-python-mlx9064x
```

Upgrade to the latest driver:

```
pip3 install --upgrade seeed-python-mlx9064x
```

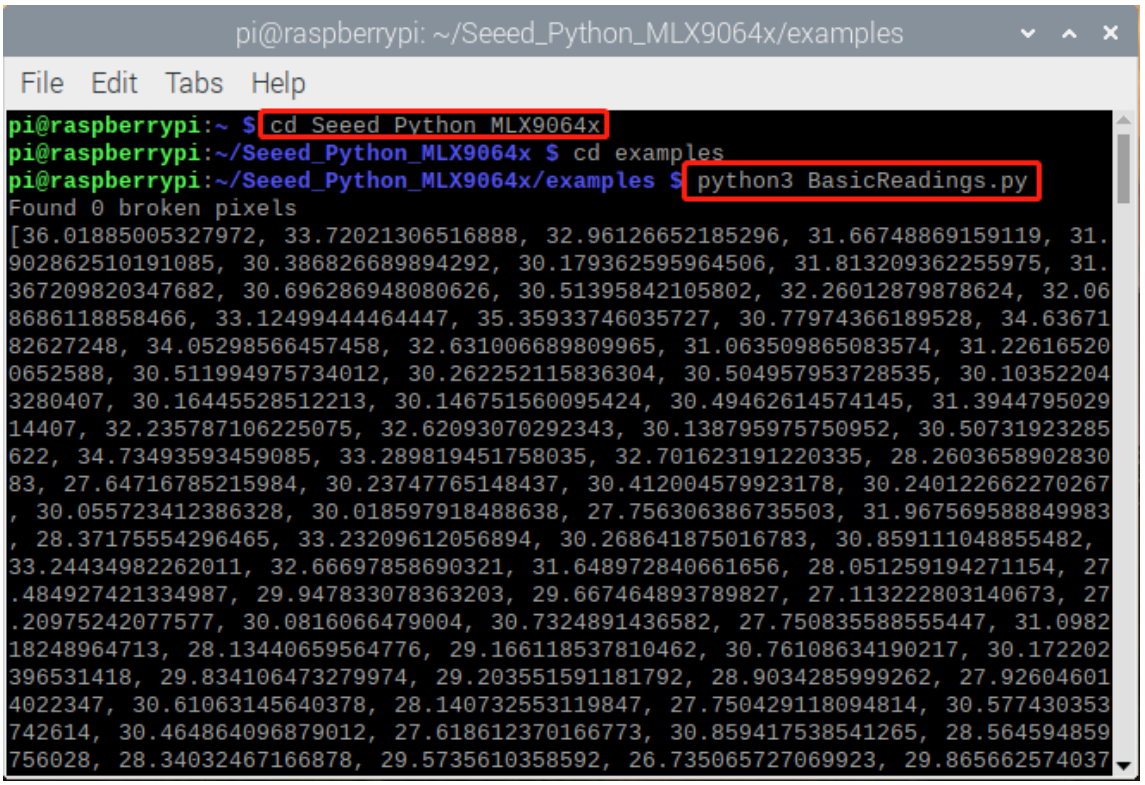
- **Step 3** Check the corresponding i2c number of the Raspberry Pi:

```
ls /dev/i2c*
```

You may get the result like this:

```
/dev/i2c-1
```

- **Step 4** Download the MLX90641 Library [https://github.com/Seeed-Studio/Seeed_Python_MLX9064x.git] by **git clone** with command.
- **Step 5** Run the **BasicReadings.py** file by the following commands:



Success
The outcome will be displayed as above if everything goes well.

Note
An upgrated UI of outcome on Raspberry Pi has been released as following:

- **Step 1** Install pyqt5:

```
sudo apt-get install python3-pyqt5 -y
```

- **Step 2** Installing from PyPI:

```
sudo pip3 install seeed_python_ircamera
```

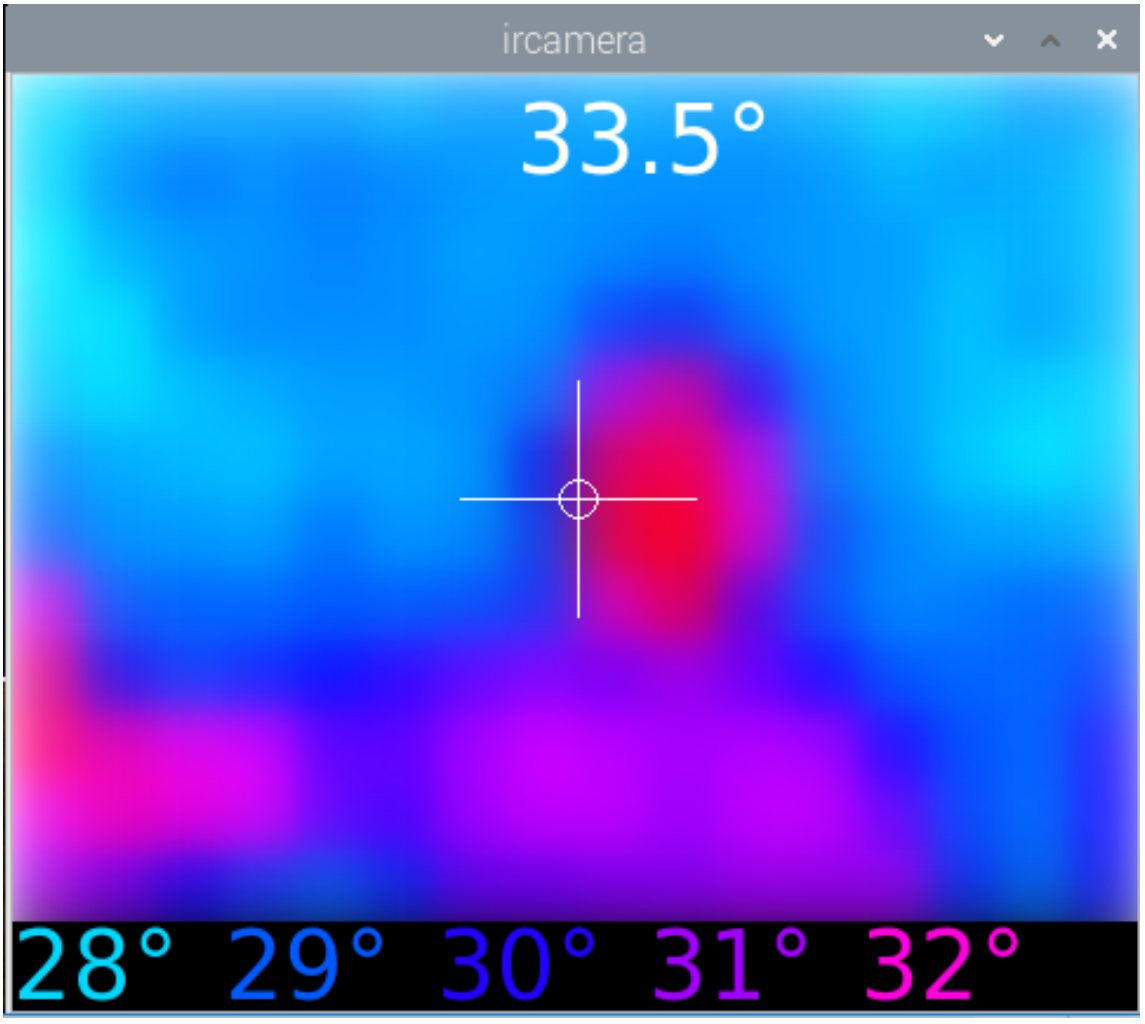
- **Step 3** Set the max i2c speed then reboot:

```
sudo sh -c "echo dtparam=i2c_arm=on,i2c_arm_baudrate=4000
sudo reboot
```

- **Step 4** Input below command in terminal:

```
sudo ircamera I2C MLX90641
```

Success
The outcome will be displayed as following if everything goes well.



Resourse

- **[PDF]** Datasheet of MLX90641 [https://files.seeedstudio.com/products/101020892/res/MLX90641-Datasheet-Melexis.pdf]
- **[ZIP]** MLX90641 Visualization [https://files.seeedstudio.com/products/101020892/res/Visualization-mlx90641.zip]

Tech Support

Please submit any technical issue into our forum [http://forum.seeedstudio.com/].

[https://www.seeedstudio.com/act-4.html?utm_source=wiki&utm_medium=wikibanner&utm_campaign=newproducts]