Evelta MLX90640 110 Degree FOV IR Array Breakout

EVE-MLX90640-WA

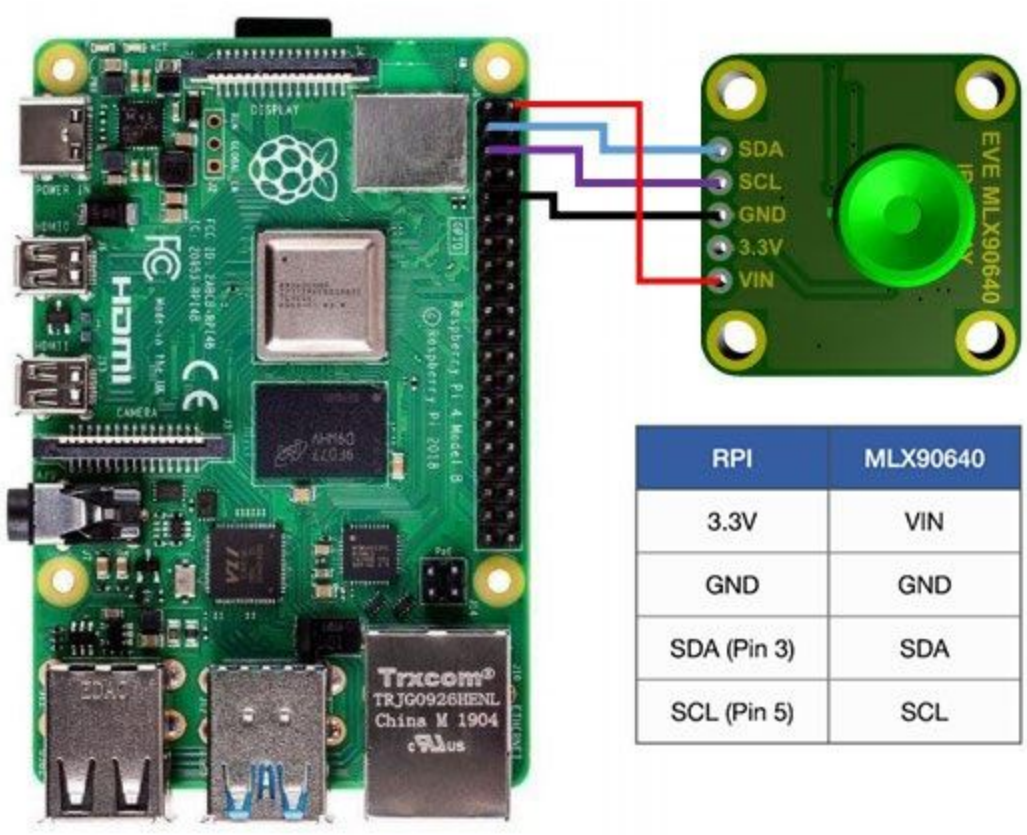
User Manual

Introduction

A thermographic camera is a device that creates an image using infrared radiation, similar to a common camera that forms an image using visible light. Instead of the 400–700 nanometre range of the visible light camera, infrared cameras are sensitive to wavelengths from about 1,000 nm (1 μm) to about 14,000 nm (14 μm). An infra-red thermal camera will enable you to explore your world in a whole new way.
This breakout board is integrated with MLX90640, a far infrared thermal sensor array (32x24 RES). It has a -40°C to 85°C operational temperature range and can measure object temperatures between -40°C and 300°C. Maintaining high levels of precision across its full measurement scale, this infrared sensor delivers a typical target object temperature accuracy of ±1°C.
It also exhibits superior noise performance.

This 32x24 pixel device is suited to safety and convenience applications that include fire prevention systems, smart buildings, intelligent lighting, IP/surveillance cameras, HVAC equipment and vehicle seat occupancy detection.

MLX90640 Breakout to Raspberry Pi board Connection



RPI	MLX90640
3.3V	VIN
GND	GND
SDA (Pin 3)	SDA
SCL (Pin 5)	SCL

The MLX90640 Breakout and Raspberry Pi communicate via the I2C protocol, which uses the hardware pins 3/5 on the Pi (SDA/SCL).

The Adafruit library will be used to read the MLX90640 thermal breakout board. Enter the following commands into the RPI terminal to ensure that the MLX90640 sensor can be visualized in Python.

```
pi@raspberrypi:~ $ sudo pip3 install matplotlib scipy numpy
```

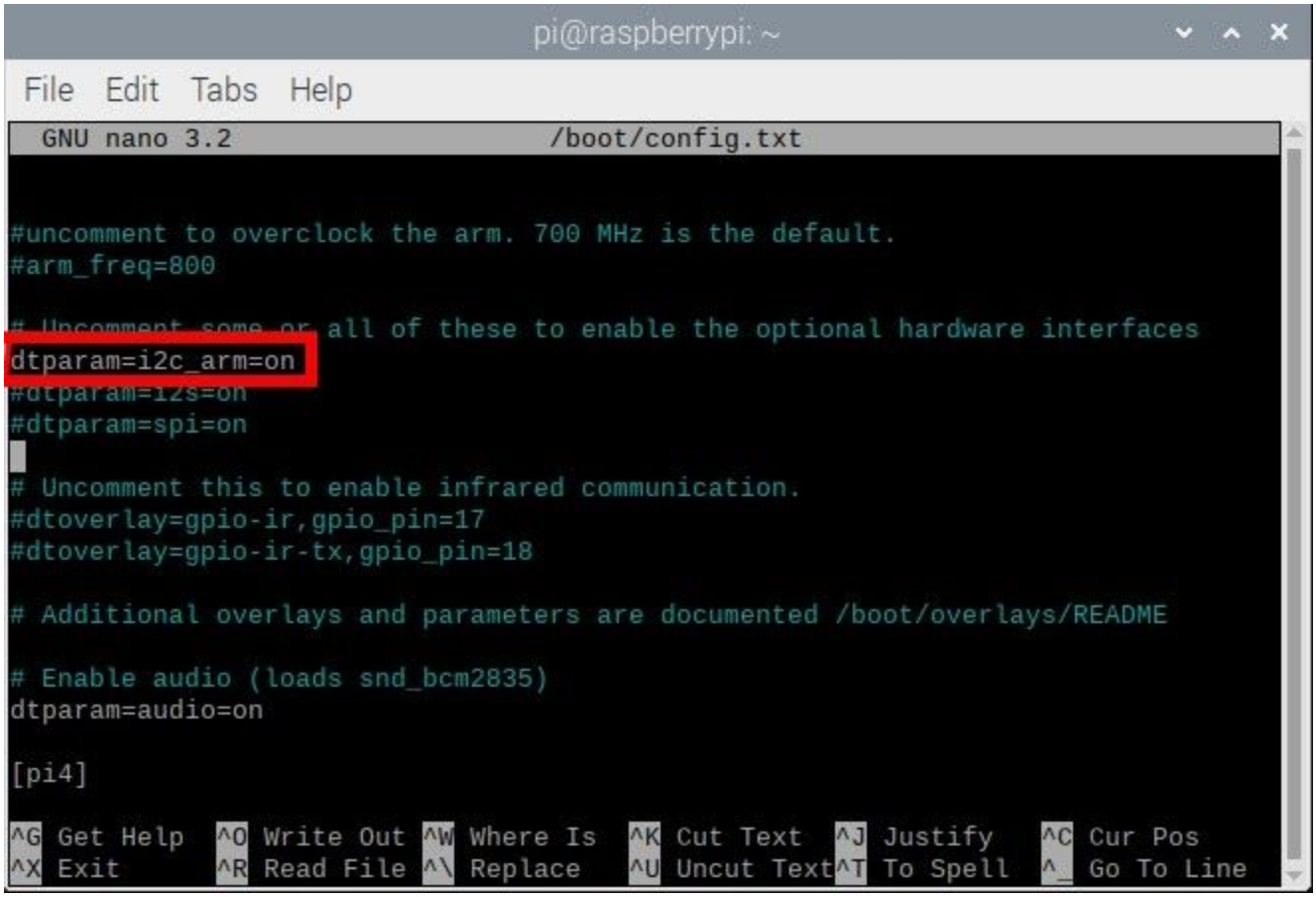
Additionally, the RPi needs I2C tools installed:

```
pi@raspberrypi:~ $ sudo apt-get install -y python-smbus
pi@raspberrypi:~ $ sudo apt-get install -y i2c-tools
```

Ensure that the I2C is enabled (via the terminal):

pi@raspberrypi:~ $ sudo nano /boot/config.txt

This command opens the boot file on the RPi. Scroll down to the dtparam=i2c_arm=on and make sure that it is uncommented:



Now I2C enabled, reboot the RPi:

pi@raspberrypi:~ $ sudo reboot

Once the RPi restarts and the MLX90640 board is wired correctly, we can check the I2C port and ensure that the RPi registers the MLX90640. This can be done with the following command:

pi@raspberrypi:~ $ sudo i2cdetect -y 1

The following output be printed on the terminal:

The number 33 printed, which is the I2C address of the MLX90640 (0x33). At this point, the MLX90640 is ready to communicate with the Raspberry Pi. Since the Adafruit library is being used, a few other libraries need to be installed:

pi@raspberrypi:~ $ sudo pip3 install RPI.GPIO adafruit-blinka
pi@raspberrypi:~ $ sudo pip3 install adafruit-circuitpython-mlx90640

Now the Python Integrated Development and Learning Environment (IDLE) is installed, but not necessarily required. An anaconda environment could also be used, but since the RPi is used here, we chose IDLE (for Python 3). IDLE, if not installed already, can be installed as follows:

pi@raspberrypi:~ $ sudo apt-get install idle3

Finally, open IDLE or Anaconda and import the MLX90640 library from Adafruit using the following test code:

```
##################################
# MLX90640 Test with Raspberry Pi
```

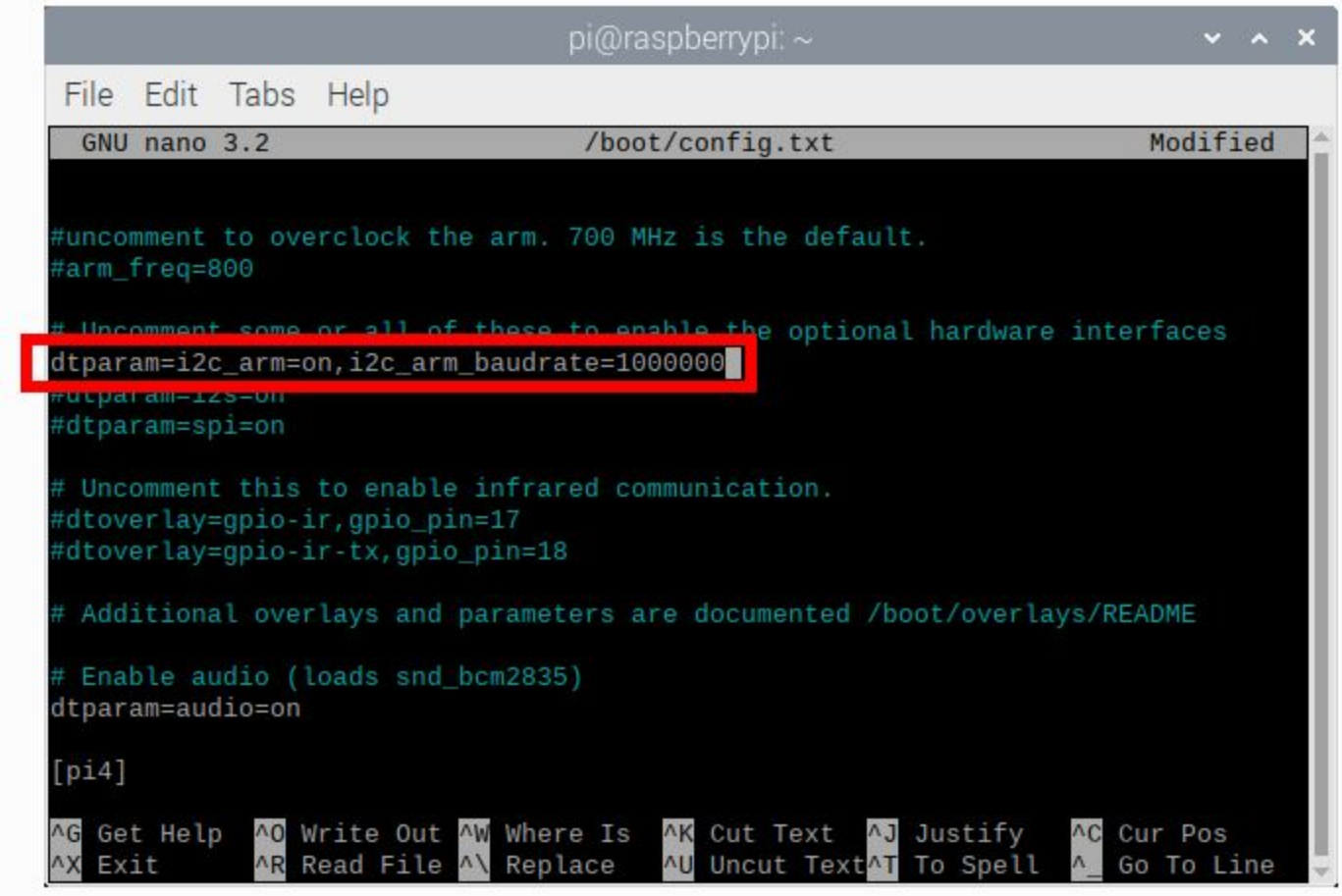
```
##################################
#
import time,board,busio
import numpy as np
import adafruit_mlx90640

i2c = busio.I2C(board.SCL, board.SDA, frequency=400000) #
setup I2C
mlx = adafruit_mlx90640.MLX90640(i2c) # begin MLX90640 with
I2C comm
mlx.refresh_rate = adafruit_mlx90640.RefreshRate.REFRESH_2_HZ
# set refresh rate

frame = np.zeros((24*32,)) # setup array for storing all 768
temperatures
while True:
    try:
        mlx.getFrame(frame) # read MLX temperatures into
frame var
        break
    except ValueError:
        continue # if error, just read again

# print out the average temperature from the MLX90640
print('Average MLX90640 Temperature: {0:2.1f}C ({1:2.1f}F)'.\

format(np.mean(frame),(((9.0/5.0)*np.mean(frame))+32.0)))
```

The code above should print out the average temperature read by the MLX90640. Pointing the MLX90640 sensor at the Raspberry Pi resulted in an average temperature of 32.5°C (90.5°F).

When reading the MLX90640, an error may appear that cites a refresh rate issue. This can be avoided by amping up the rate of the I2C device on the RPi. To do this, we need to change the following back in the ‘config.txt’ file:

pi@raspberrypi:~ $ sudo nano /boot/config.txt

Scrolling down to the uncommented ‘dtparam=i2c_arm=on’ - we also want to add the following line that increases the I2C speed to 1Mbit/s:



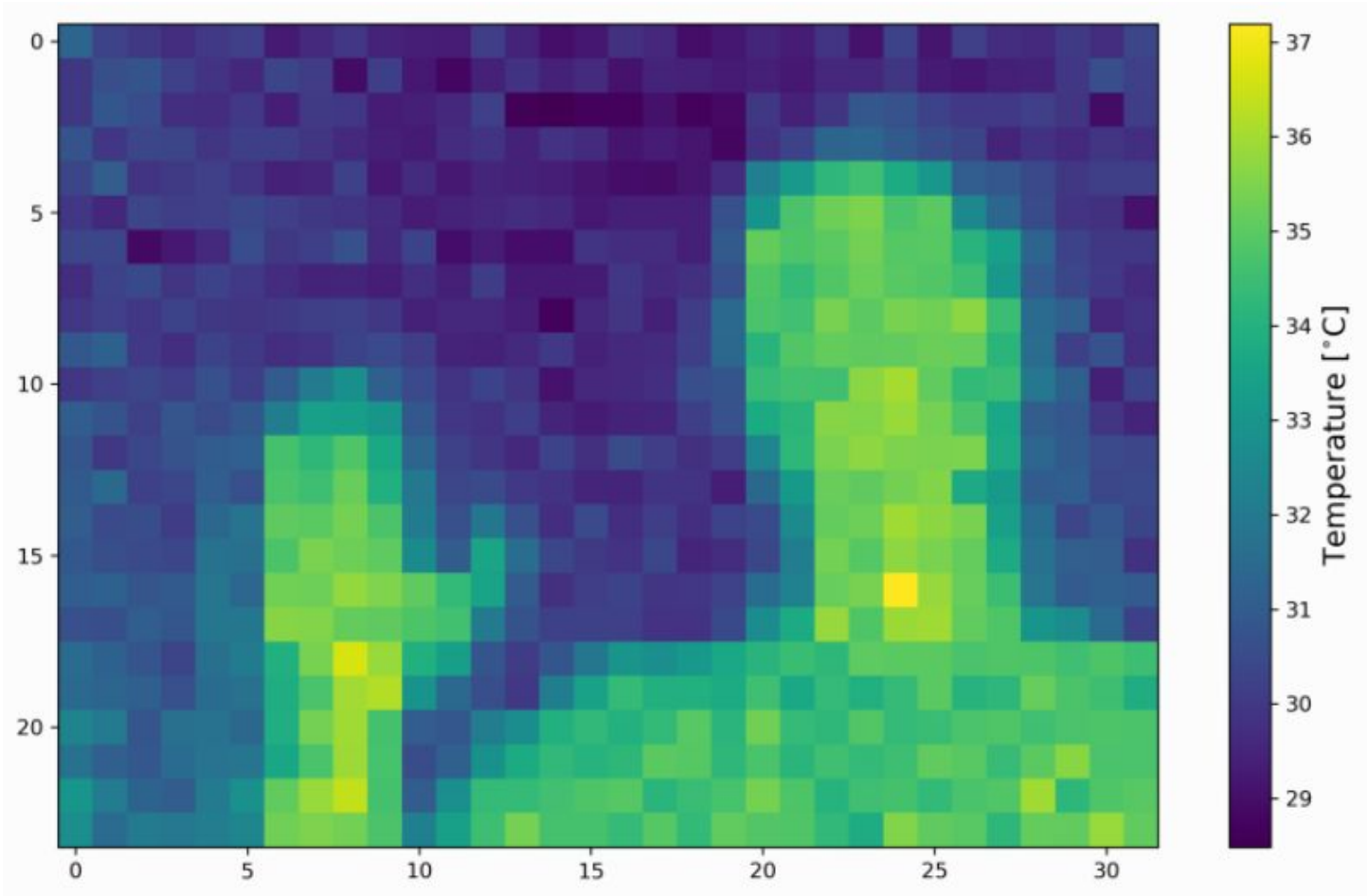
Be cautious when increasing the I2C baud rate above the recommended speed (400kbit/s). This high speed can cause overheating of the Pi, so ensure that the board is properly ventilated or actively cooled. In an upcoming section, some routines for plotting the 24x32 temperature grid will be introduced, where this 1Mbit/s will be important for creating a near real-time thermal camera with the MLX90640 sensor.

A simple implementation of the MLX90640 visualization is shown below using ‘imshow’ in Python, with the left-right flipping done in the code:

```
##########################################
# MLX90640 Thermal Camera w Raspberry Pi
# -- 2Hz Sampling with Simple Routine
##########################################
#
import time,board,busio
import numpy as np
import adafruit_mlx90640
import matplotlib.pyplot as plt

i2c = busio.I2C(board.SCL, board.SDA, frequency=400000) #
setup I2C
mlx = adafruit_mlx90640.MLX90640(i2c) # begin MLX90640 with
I2C comm
mlx.refresh_rate = adafruit_mlx90640.RefreshRate.REFRESH_8_HZ
# set refresh rate
mlx_shape = (24,32)

# setup the figure for plotting
plt.ion() # enables interactive plotting
fig,ax = plt.subplots(figsize=(12,7))
therm1 = ax.imshow(np.zeros(mlx_shape),vmin=0,vmax=60) #start
plot with zeros
cbar = fig.colorbar(therm1) # setup colorbar for temps
cbar.set_label('Temperature [$^{\circ}$C]',fontsize=14) #
colorbar label

frame = np.zeros((24*32,)) # setup array for storing all 768
temperatures
t_array = []
while True:
    t1 = time.monotonic()
```

```
    try:
        mlx.getFrame(frame) # read MLX temperatures into
frame var
        data_array = (np.reshape(frame,mlx_shape)) # reshape
to 24x32
        therm1.set_data(np.fliplr(data_array)) # flip left to
right

therm1.set_clim(vmin=np.min(data_array),vmax=np.max(data_arra
y)) # set bounds
        cbar.on_mappable_changed(therm1) # update colorbar
range
        plt.pause(0.001) # required

fig.savefig('mlx90640_test_fliplr.png',dpi=300,facecolor='#FC
FCFC',
                    bbox_inches='tight') # comment out to
speed up
        t_array.append(time.monotonic()-t1)
        print('Sample Rate:
{0:2.1f}fps'.format(len(t_array)/np.sum(t_array)))
    except ValueError:
        continue # if error, just read again
```

The code above should output an image similar to the following:


0
5
10
15
20
0
5
10
15
20
25
30
37
36
35
34
33
32
31
30
29
Temperature [°C]