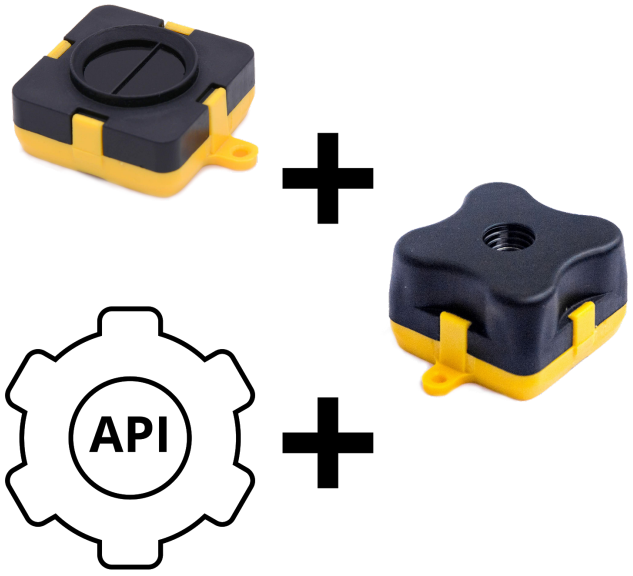
Fever Screening Applications

Quick Guideline



Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

Table of contents:

Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

1. Introduction

The purpose of this document is to guide the user of the Terabee Fever Screening Kit to create a quick fever screening device. This document describes the components needed and the steps to follow. This solution intends to address a quick and cheap version of fever screening solution.
This document is a guideline and needs to be considered as a goodwill gesture from Terabee to users.

2. Components needed

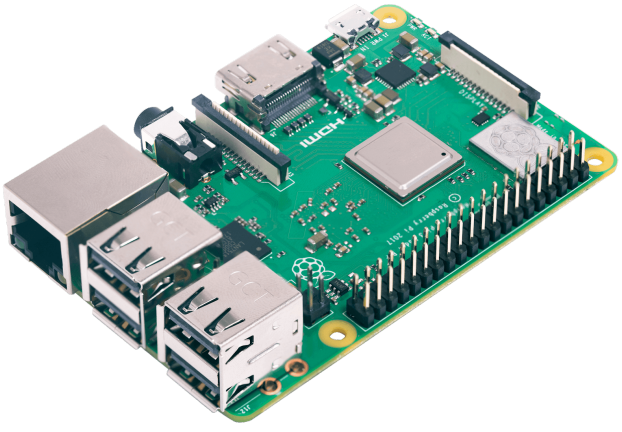
In this section you will find the components that we suggest you use.

2.1. Terabee Fever Screening Kit

The Terabee Fever Screening Kit is an easy solution made with the Evo Thermal 33 and the Evo Mini sensors. The Terabee Fever Screening Kit contains source code which allows quick development of fever screening applications. This software (API in Python) is oriented to help/support or boost your fever screening application saving you weeks of testing and development. Link to purchase the sensor: https://www.terabee.com/shop/covid-products/terabee-fever-screening-kit/

2.2. Raspberry Pi 3B+

You will need a companion computer/PC to run our API software.
We recommend the famous Raspberry PI 3B+ for easy access, sufficient computation power and low price. (other models or companion computers can be also used)



Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

2.3. Display

There are multiple options for a display. Raspberry Pi 3B+ allows easy connection to all kinds of display devices, from text-only LCDs to HDMI screens (3.5" - 7"). Users can connect any type of display compatible with the Raspberry Pi 3B+ or companion computer.

2.4. Connection elements

Terabee sensors have a USB interface. Two USB cables are needed to connect the sensors to the companion computer.

Raspberry Pi 3B+ needs to be connected to the power supply via micro USB port. A power supply adapted to the Raspberry Pi is required.

Different displays have different ways to connect to the Raspberry Pi 3B+: HDMI, USB, GPIOs, DSI, etc. Please pay attention to the specific connection between the display and the Raspberry Pi 3B+.

2.5. Indicative Price Table

Component	Indicative Market Price
Terabee Fever Screening Kit	290 €
Raspberry Pi 3B+	35 €*
Cables (USB, HDMI, etc)	10 €*
Display	30 €*
Total cost	**365 €***

**This is an approximate price. The cost of each component may vary depending on geolocalization, type, model, quality, etc. Shipping costs are not included in the price estimation.*

Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

3. Raspberry Pi 3B+ setup

This section describes our recommended steps needed to set up the Raspberry Pi 3B+.

1. Download the latest image from the official website here: https://www.raspberrypi.org/downloads/raspberry-pi-os/
2. Follow the installation instructions on the Raspberry Pi website: https://www.raspberrypi.org/documentation/installation/installing-images/README.md
3. Follow our software guidelines below.

Note: the newest versions of Raspberry Pi OS come with Python 3 pre-installed

Software

The source code is written in Python 3 in the form of an importable API and can be downloaded from our website under the downloads section:

https://www.terabee.com/shop/covid-products/terabee-fever-screening-kit/

The user can simply import our *FeverDetection* thread object and initialize it.

System requirements

- Python >= 3.6
  - pyserial~=3.4
  - crcmod~=1.7
  - opencv-contrib-python
  - numpy~=1.18.2
- 150 MB of RAM
- User requires permission to access serial ports. A simple guide can be found here.

Python3 dependencies

If there are missing dependencies, they can easily be installed using pip and the requirements file, as follows:

1. If both python 2 and python 3 are installed on your machine, use the following:

```
pip3 install --user  -r "requirements.txt"
```


Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

2. If you have only python3 installed on your machine, use the following:

```
pip install --user  -r "requirements.txt"
```

Usage

Initialization

Start by importing the fever detection object and the APIevent enum to your software. The fever detection API runs as a separate thread, you will need to feed it an event to stop it. It also takes the ports, temperature unit (C or F) and optionally, a callback function for when an event is registered as keyword parameters. The *APIevent* is used by the callback to notify you of events. Example usage:

```
from fever_detection import FeverDetection
from multiprocessing import Event
from API_events import APIevent

stopEvent = Event()
MyDetector =FeverDetection(portnames={'evo_thermal_port':"/dev/ttyACM0",
                           'evo_mini_port':"/dev/ttyACM1"},
                           settings={"unit": "C"},
                           stopEvent=stopEvent,
                           callback=callback)
```

Initializing the sensors

After the object is created, its method *initialize()* needs to be called. This will start the sensor processes and initialize the internal state machine. Example:

```
MyDetector.initialize()
```

Starting the system

Since this is a Thread object, you will need to call its *start()* method in order to run it. After this, it now runs in an infinite loop until the stop event is received.

```
MyDetector.start()
```


Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

Receiving the data

There two ways to get data out of the system. You can either poll the *getStatus()* method or initialize it with a callback function and work with that. Of course, you can also combine the two methods. Both the callback and the *getStatus()* method will return a string that looks like this: ***Current state: Alignment, Current distance: 56.0, Current temperature: 0.0***. However, there is one difference: the callback will feed you an *APIevent* such as *PERSON_DETECTED* or *SYSTEM_READY* since its purpose is to provide feedback on events. Whereas the *getStatus()* method will return the current state of the state machine. The *event* argument taken by the callback is the triggered *APIevent* and the *temperature* is of type *float*. Example usage:

```
# Polling method
MyDetector.getStatus()

# Callback method
def callback(string, event, temperature):
    print("In main thread callback function. Input: ", string)
    if event == APIevent.RESULT_READY:
        print("Current temperature", temperature)
```

Detection failure

In the event that the thermal cannot get a temperature reading, the API will notify with the callback, sleep for 1 second and then reset the state machine.

Stopping the system

In order to stop the API, you will have to set the stopEvent to True and then call the *join()* method of the thread. Example:

```
stopEvent.set()
MyDetector.join()
```

Quick start script

All the example code above is located in the test_script.py file and can be run immediately after all the dependencies are installed. You will need to modify the ports

Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

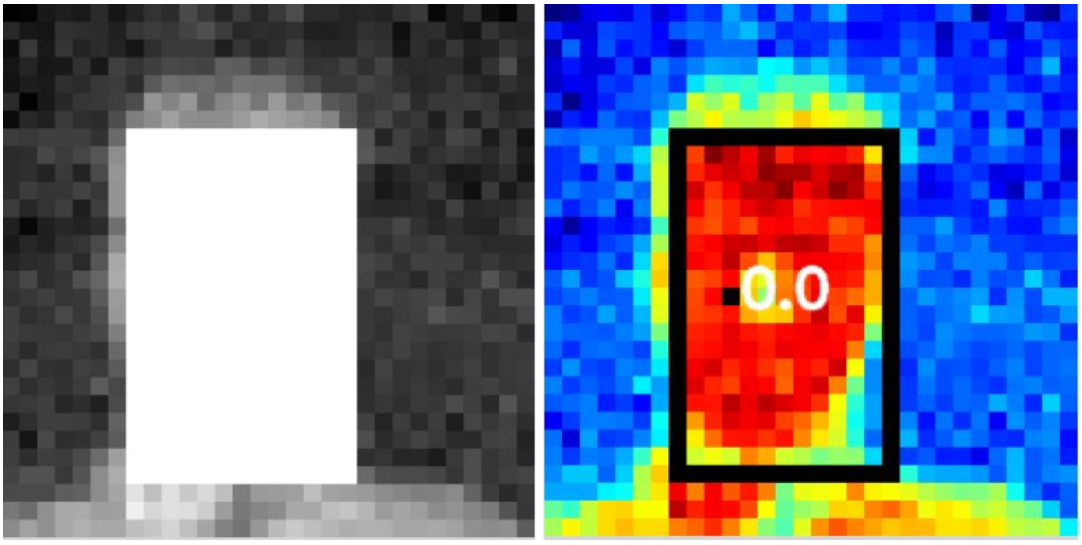
to the ones that correspond to the Evo Mini and Evo Thermal 33 on your setup (see Initialization). After modifying the ports, run the script by typing in:

```
python3 test_script.py
```

This sample code will print out messages in the console.

Debug mode

If you would like a visual guide, you can turn on *debug* mode for the Evo Thermal. This will open up two windows, one called “Grayscale” and one “Heatmap” as well as providing more text information in the console. This is what the Evo Thermal is seeing. In these images you will be able to see the algorithms at work. The “Grayscale” image will highlight the scan area by completely masking it, while the “Heatmap” will display just the rectangle around that area and the detected temperature of the person.

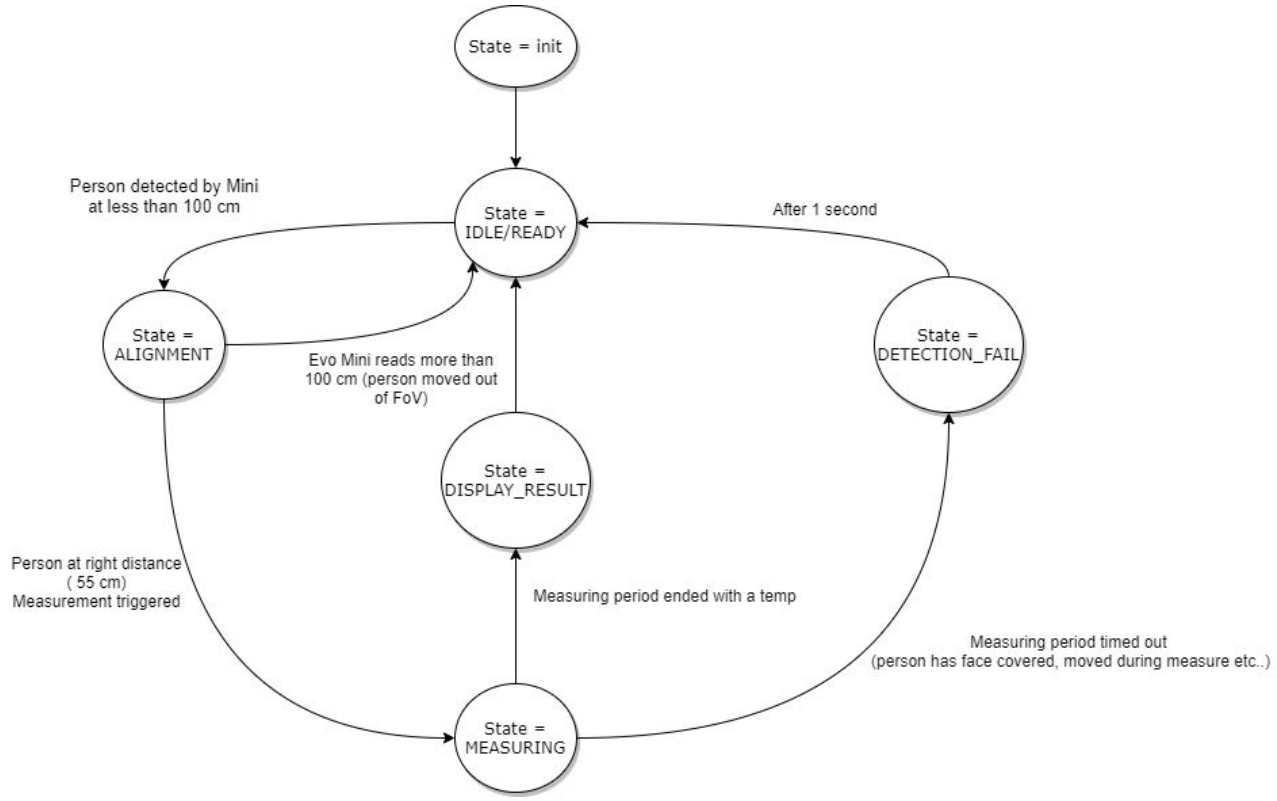


To turn *debug* mode on, simply edit the ***fever_detection.py*** file, line 77 and set debug to True, as follows:

```
self.evoThermal=EvoThermal(
    portname=self.portnames["evo_thermal_port"],
    unit=self.settings["unit"],
    shared_variables={'trigger_event':self.distanceTrigger,
    'temperatureReady': self.temperatureReady,
    'temperatureQueue': self.temperatureQueue},
    debug=True
)
```


Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)

State machine diagram



Terabee, 90 Rue Henri Fabre
01630, St Genis-Pouilly, France (next to CERN)