PITA Report

Alice Henman
email: alicehenman@icloud.com

September 19, 2025

Abstract

PITA is a multi-measurement device that can take readings in temperature, voltage and current. The device is programmed to take measurements both via an infrared camera and through an analogue front end (AFE) to be stored on a microSD card or pushed to an external webpage. PITA aims to be a cheaper and highly portable alternative to a traditional infrared camera while offering more versatility in its capabilities. Arduino IDE and Visual Studio Code have been used to write the C++ that dictates PITA, as well as a small amount of HTML and JavaScript for the configuration settings page.

1 Introduction

The idea of The Portable Infrared Temperature Analyser (PITA) was first realised by my supervisors[1] as a solution to measuring the temperature of something in a room that is uninhabitable by humans.
Following the integration of an upgraded capacitor discharge system for POPSB[2]; safety test measurements were required on the newly installed resistors to ensure they weren't reaching dangerous temperatures. However, traditional contact sensors would be at risk of electrical coupling and expensive damages and the high-voltage environment of the capacitor room does not allow for human presence, so a new innovative solution was required in the form of an Arduino with an infrared camera zip-tied to a battery pack. From here PITA was born and developed into what it is today: A portable multipurpose measurement device with the aim to be cheaper and more accessible than other infrared camera's on the market.

2 What is PITA?

As shown in Figure 1, the most critical component is the ESP32-S3 micro controller, which stores the code that dictates how PITA works in its internal flash memory and communicates with all of the components to tell them what to do. The measurement components consist of a 60x80 FLIR Lepton infrared camera[3] for non-contact thermal imaging; an NAFE13388 (NXP analogue front end) with 2 connected 3.5 audiojack ports which act as probes, PT100's can be connected to these probes for contact temperature measurements; not visible is another temperature sensor for measuring the ambient temperature, this has not yet been integrated.
For data storage, PITA includes a microSD breakout board that allows users to store data offline using their own microSD card, and the option to push measurements to an online database that can be accessed via the PITA Webpage.
The final hardware component is a button that will trigger the "configuration mode".

3 How PITA works

3.1 Configuration mode

When the configuration button is pressed "configuration mode" will be triggered in one of two ways. If PITA is offline, the ESP32 will set up an access point [2] to which the user can connect to as they would any WiFi network, then by entering the PITA device's IP address into a browser the

[1] Arnaud Bessonnat and Marco Savina
[2] The power converter for the booster
[3] Though we plan for the optional integration of a higher resolution FLIR camera, notably 120x160

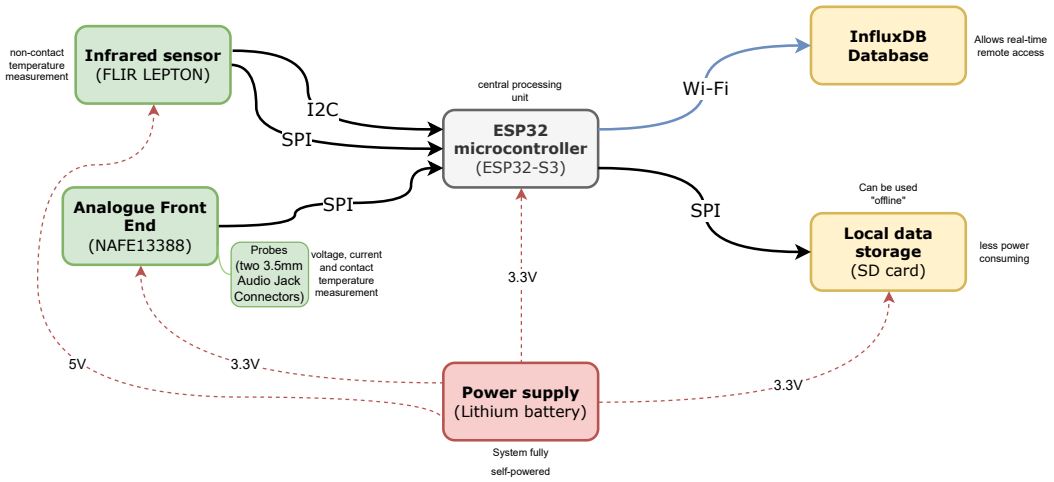

Figure 1: A functional diagram showing all of the system components.

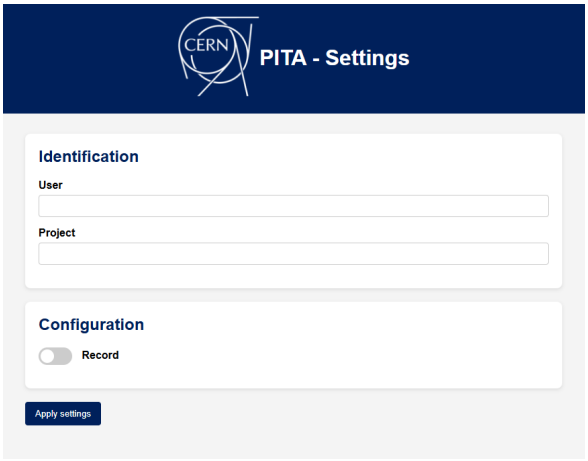

Figure 2: Configuration Settings via the ESP32 access point.

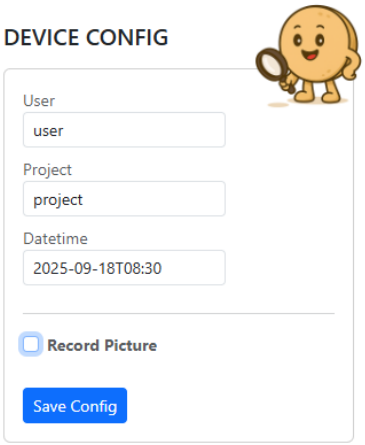

Figure 3: Configuration settings via the PITA Webpage.

configuration settings will be loaded; this can be done on a computer, tablet, phone, or any other device. However, if PITA has online access, this button press will simply erase the username of the previous handler allowing full access to the device via the PITA Webpage including the option to change the configuration settings [3], and add a new username to lock access and protect the user's project from unauthorised interference.

Here one can toggle "Record" on to set the configuration and start taking measurements[4]. Such settings include enabling sleep mode to pause the CPU and WiFi communication between measurements to conserve power; choosing how often to take readings; selecting the storage type for online or offline storage or a combination of both; enabling or disabling the infrared camera, one or both of the probes and the type of measurement they will be performing; and enabling the ambient temperature sensor. Then by applying settings the ESP32 updates its internal settings and restarts to overwrite its old configuration.

3.2 The Results

If the data is to be held in local storage the ESP32 will create a new folder on the SD card at setup with the name of the project, users can also specify a filename to personalise their measurements within the project. Probe and ambient readings will be stored as single text files for easy comparisons with a timestamp, followed by what was being measured (for the probes only) and the reading taken, infrared measurements will be stored in a new folder labelled FLIR_Readings as each measurement will save a csv file of the raw values from the camera (in centikelvin), a text file with the maximum and minimum temperatures recorded in that frame and a jpg image to

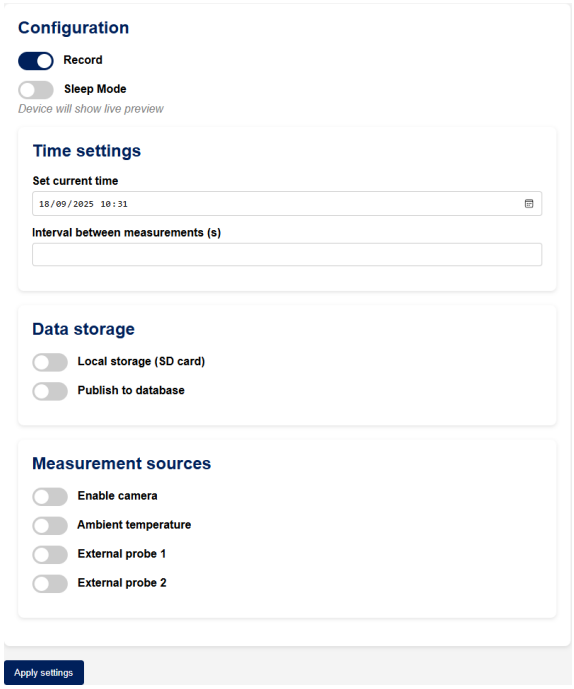

Figure 4: Extended configuration settings via the ESP32 access point.

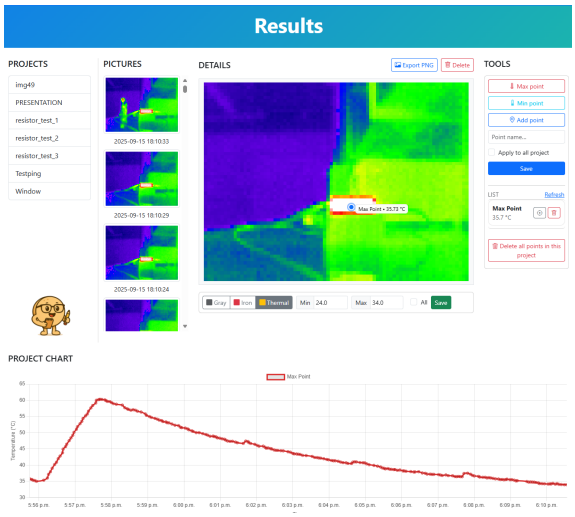

Figure 5: Results page of the PITA Webpage.

visualise the frame.
Alternatively, publishing results to the online database will push the readings to the PITA webpage as they are taken. Figure [5] shows the layout of the online database. On the left are the list of projects belonging to the user and within each project the measurements are stored in chronological order, the Webpage's interactive interface allows the user to add multiple points to any of the images and apply it across all of the captures in the project to create a plot as seen at the bottom of the page, making it easy to visualise trends and make comparisons.

3.3 Live mode

The PITA Webpage has another prominent feature unofficially titled "live mode"[6]. In the devices tab all of the registered devices are shown on the left as well as the user they are registered to, as discussed above, pressing the configuration button will free up the device allowing anyone to view the measurements and change the settings. The live preview in the center posts a new frame from the FLIR Lepton camera every 1 second to ensure it's working and recording the intended field of view. There is also the option to save the current live frame to a project or export it. As previously stated, the online configuration settings are also found on this page.

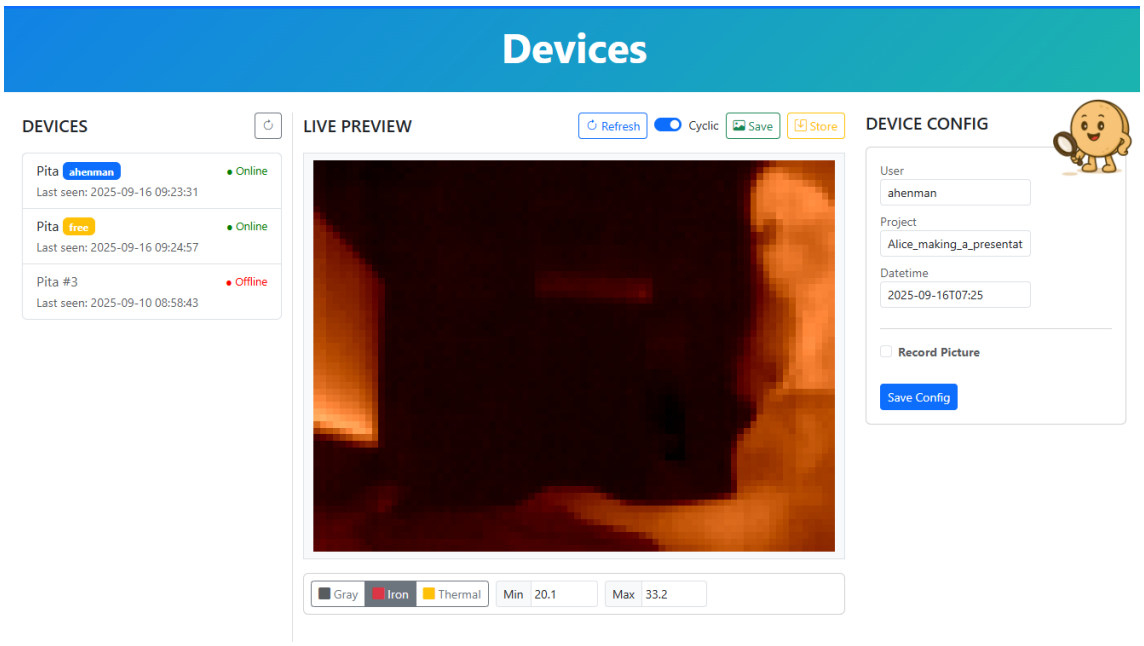

Figure 6: Devices page of the PITA Webpage.

4 The Code

I used C++ to write the majority of my code, with the only exception being the small amount of HTML and JavaScript I used for creating the Configuration Page accessible via the ESP32's access point. C++ was the best language for this project as all code is compiled to machine code prior to being uploaded, this increases the speed and efficiency at which code is ran on the ESP32 which is crucial to PITA's design; this and the fact that many of the libraries that were required are written in C++ made it the obvious choice. I began using Arduino IDE at the start of my project, but due to the increasing volume of code I had and the time it was taking to upload each time. I switched to VSCode for the faster upload speed and so that I could log all of my changes on GitLab, making it easier to revert to old save files in the chance something went wrong.

4.1 For the FLIR Lepton camera

Although there is a lepton library for Arduino, and this is what I was using originally, I transitioned to using raw SPI (Serial Peripheral Interface) to retrieve the frames. This was done to optimise the speed of acquisition(higher frame rate and less "cutting out") when implementing the live feed so it is more appealing to the user and therefore more marketable. In setup I communicate with the camera via the I2C (Inter-Integrated Circuit) protocol to ensure it is booted and ready to start taking pictures and then I run an FFC (Flat Field Correction) normalisation to prevent drifting[4]. The SPI is what actually retrieves the image by checking the header of each of the incoming packets[5] to decide whether to discard it or if not whether it is the first packet of an image and only moves on once it has found this first packet. From here it will scan for the second, etc. until the whole image has been compiled. For SD storage I have another function that scans each value in the buffer to find the maximum and minimum values in each image, then scales the raw values so they can be applied to a colour scale[6] in order to make the JPEG image.

4.2 For the Analogue Front End

This component was by far the most challenging to implement and required the consultation of numerous data sheets in order to construct the code. The NAFE13388 is an extremely versatile tool but by consequence provides to be equally as complicated; furthermore, it was difficult to find code examples in the language I was using that matched my purpose. In setup, I configure several different channels on the AFE to handle the presence of 2 different probes as well as the several different measurement modes. Each channel essentially has a different configuration which take

[4] temperatures becoming less accurate over time due to internal heating of the components.
[5] Each image is to be stored in a buffer which is made up of 60 packets *depending on resolution.
[6] I have used turbo.

measurements from different ports on the AFE, use a different gain for the PGA (Programmable Gain Amplifier) and use internal voltage/current excitation, etc.. Each time a reading is taken, the code accesses the channel corresponding to the correct probe and obtains a raw measurement which I can then extract the desired measurement from. The general process goes as follows; firstly, the raw 24-bit value extracted from the AFE needs to be sign extended from two's compliment to a signed integer by equation (1), the voltage can then be calculated from here via equation (2). Depending on what measurement is required equation (3) is used to obtain current or resistance, which is then used to derive the temperature via PT100 resistance-temperature conversion tables. For ease, I constructed a quartic equation which matches the table well for my temperature values of interest to avoid storing another large buffer on the ESP32's RAM.

$$sINT24 = (raw + 2^{23}) \mod 2^{24} - 2^{23}, \tag{1}$$

$$Voltage = \frac{10 \times raw}{2^{24} \times GAIN}, \tag{2}$$

$$R = \frac{V}{I}. \tag{3}$$

Where raw is the raw AFE reading; sINT24 is the signed integer representation of the raw value; GAIN is the amplification factor applied to the signal before it reaches the ADC (analogue to digital converter); R is the resistance, I the current and V the voltage.

4.3 For the SD card

I used the SDFat library in order to add timestamps to the readings, this is done by first checking if PITA can connect to the WiFi; if so, it uses the Network time protocol (NTP) to synchronise PITA's internal clock over the internet and get the true local time. If there is no WiFi connection it will instead use the datetime entered in configuration mode and start the internal clock from there. At setup a folder for the project is created and within it a folder or file for each parameter being measured; on the first reading the index of the last uploaded reading (if it exists) is checked so the file indentation can carry on from where it left off. I have protected against crashing by checking first if the SD card exists and if not local storage is disabled, additionally, before any folder or file is created the program checks if it already exists to prevent creating many copies of the same folder or accidentally overwriting files.

4.4 For the WiFi communication

I originally used the ESPAsyncWebServer in combination with AsyncTCP to handle requests from the PITA webpage and for setting up the ESP32 access point, this works well for smaller projects as it runs in the background to prevent the code from blocking while it handles requests. However, because the buffers to store the raw lepton images occupy a lot of RAM, the code was struggling to push this data to the database using HTTP while it was simultaneously sending the same large files in the background for the live view. Because PITA should be able to handle continuously sending images to the webpage on request and posting data to the online database at the same time, I opted for switching to the regular WebServer library that handles sending one Json file at a time. This seems rather contradictory at first, but unless the RAM is upgraded it is unbeneficial to attempt to execute large data transfers similarly on the ESP32.

4.5 For the Configuration

I assigned a digital interrupt pin to the configuration button, so that when pressed it will set the bool ConfigMode to true, wake up the device if it was asleep and set sleepmode to false so PITA can not reenter sleep during the configuration set up. Then, in both the setup and in the loop if ConfigMode = true is detected PITA will respond accordingly. If there is no WiFi connection it will create the access point, and this will pause all measurements until a new configuration has been saved after which the ESP32 will restart to implement these changes, or it times out. If PITA was already in live mode then the button press simply logs the old user out and allows the new user access to the configuration settings on the devices tab of the PITA Webpage and enter their own user details.

5 Conclusion

The latest PITA prototype is working remarkably well for the short amount of time I have been working on and improving it, and this project has been an overall very rewarding experience. Future improvements I believe need to be implemented include adding more RAM to the ESP32, adding probe data storage options to the online database, officially adding a means for ambient temperature detection and optimising PITA-Webpage interactions. In terms of extra RAM I believe this can be done by using an ESP32 which comes with added PSRAM, then it will be able to handle the large buffers filled with raw lepton data more smoothly and create the possibility for using a higher resolution camera in the future, the code is already built to handle variable resolutions. There are many options for improving the communication between PITA and the Webpage that I have not had time to test, one of which is streaming the information in binary, other methods involve opening up a websocket connection to the server, or perhaps with a larger RAM we can safely switch back to using Async.

6 Data sheets

https://cdn.sparkfun.com/assets/8/0/d/b/3/DS-16912-FLiR_Lepton_-_Breakout_Board_V2.pdf
https://www.mouser.com/pdfDocs/UM12203.pdf?srsltid=AfmBOopZdrfcjUuFDiQq3lqoFeBNkq7fhJH8SA2FsdatzyM
https://www.nxp.com/docs/en/data-sheet/NAFE13388.pdf
https://www.nxp.com/docs/en/application-note/AN14539.pdf