Master of Aerospace Engineering

Master thesis

AI-driven payload for Miniaturised spacecraft

Author: Maanasa Sachidanand

Supervisor: Prof. Andreas M. Hein

Tutor: Prof. Arnaud Dion

September 21, 2022

TABLE OF CONTENTS

LIST OF FIGURES

ACRONYMS

AI Artificial Intelligence. 1, 6, 9–12, 14, 16, 18, 21, 30, 32–36, 38, 40–43, 49–52

CNES Centre National d'Études Spatiales. 8

CNN Convolutional Neural Network. 16–18, 52

COTS Commercial off-the-shelf. 9, 12, 30

EGSE Electrical Ground Support Equipment. 31

FET Field-effect transistor. 30

INCOSE International Council on Systems Engineering. 23

IPC Interprocess communication. 42

IR Infrared. 4, 23, 24, 30, 32, 35, 48

ISS The International space station. 9

MBSE Model-based systems engineering. 10, 21, 23, 50

MPSoC Multiprocessor system on a chip. 1, 6, 49, 51, 52

OS Operating system. 31, 32

PCB Printed Circuit Board. 47, 48

PCIe The Peripheral Component Interconnect. 14, 30

PNG Portable Network Graphics. 32, 35

RBM Restricted Boltzmann Machine. 16

RNN Recurrent Neural Network. 16

UART Universal Asynchronous Receiver/Transmitter. 30, 46

Abstract

Typically, A spacecraft's health is monitored by sending telemetry data to the ground segment. This can create latency and data limitations in identifying sources of electrical error or malfunction on electronic devices. Early fault detection of onboard electronics of a satellite can ensure safe operation under nominal conditions. In this thesis, Artificial intelligence techniques are proposed for a compact infrared payload that monitors the satellite's electronics. The payload uses infrared thermography which is a non-invasive method to monitor faults due to temperature variations on the electronic boards and it is presently used in several industries such as agriculture, manufacturing etc. Thus, the different AI anomaly detection techniques for detecting faults on thermal images are studied. These algorithms are implemented and tested on the first configuration of the payload which includes the use of Raspberry Pi Zero 2W as the onboard computer. The AI techniques also show high accuracy on simulated faults for the space payload. Thus, this thesis contributes to the development of a compact infrared payload that demonstrates the potential of edge artificial intelligence in space. It also presents the challenges and limitations of the deployment of AI techniques for processing thermal images on the edge. The thesis concludes with the advantages of the use of MPSoC as an onboard computer for the second configuration of the payload.

Keywords:

Satellite payload · Artificial Intelligence · Edge AI · Health monitoring · Thermal anomaly detection

Acknowledgments

I take this opportunity to offer my sincere gratitude to the people who helped me in completing this Master thesis.

I would like to thank my supervisor, Prof. Andreas Hein, for choosing me for this wonderful project and for offering me the support and resources to perform the different tasks required in the internship. His feedback and guidance were important in writing the thesis. He has imparted system-level thinking and knowledge which will guide me towards becoming a good engineer.

I would like to thank the University of Luxembourg for providing labs and hardware. The summer party and other networking events conducted by SnT have been valuable for my professional growth. It helped me meet researchers, professionals and industrialists from different educational backgrounds.

I would like to thank my family and friends for their consistent support and belief in me. They have cheered me to grow into a mature individual and become a successful engineer. They have been with me during tough days and guided me towards positive thinking.

I would like to thank SpaSys group members for their feedback, support and guidance on my master thesis. The conversations about life and love for aerospace will always remain with me. Special thanks to my team members Dr.Jan thoemel and Konstantinos Kanavouras for their guidance and input in the development of the payload and the intellectual discussions about the space mission.

I am grateful to ISAE-SUPAERO for providing me with all the knowledge that was essential to completing the thesis. I would like to thank Prof. Ahlem Mifdaoui and Prof. Arnaud Dion for imparting knowledge in the field of embedded systems. This knowledge was essential during my thesis. Their feedback and guidance have improved my technical skills as an engineer.

Thank you all very much for believing in me.

CHAPTER 1

INTRODUCTION

Fault detection and analysis of satellites plays an important role in increasing the quality and reliability of space missions. This is important due to the high cost of space missions and the irreparability of satellites in orbit. The satellites in orbit are faced with a harsh environment due to radiation effects, solar flares, cosmic rays, plasma etc [4]. Hence, the onboard electronics are affected by ionization events, single event effects and spacecraft charging which can cause a sudden failure in the system [5]. These events can lead to degradation in performance or complete functional failure of the satellite. Thus, the early detection of faults in space electronics hardware during in-orbit operation can ensure the early diagnosis of faults by ground operators and also record the faults for prevention in future missions.

1.1 Traditional technology versus Edge Computing

The most commonly used fault detection methods involve sending the telemetry information of bus current, battery voltage, and temperature for detection of faults to the ground station. This can be observed in CNES spacecraft which use telemetry information to monitor a set of critical parameters [6]. The earlier methods of health monitoring involved setting thresholds and finding parameters which are outside the normal range of operations as well as long-term supervision of daily statistical features. These methods were followed by the use of machine learning algorithms to detect anomalies in telemetry data. However, telemetry data latency is observed due to the large distance of the satellite from the earth's surface and duration until the availability of a line of sight. This is represented in Fig. 1.1 where the satellite is unable to transmit telemetry information to the ground station antenna if it is not within the line of sight of the Antenna. Other limitations include loss of information during transmission of telemetry data to earth and also failure to react immediately in case of anomalies. These factors demand new methodologies to detect anomalies in spacecraft.

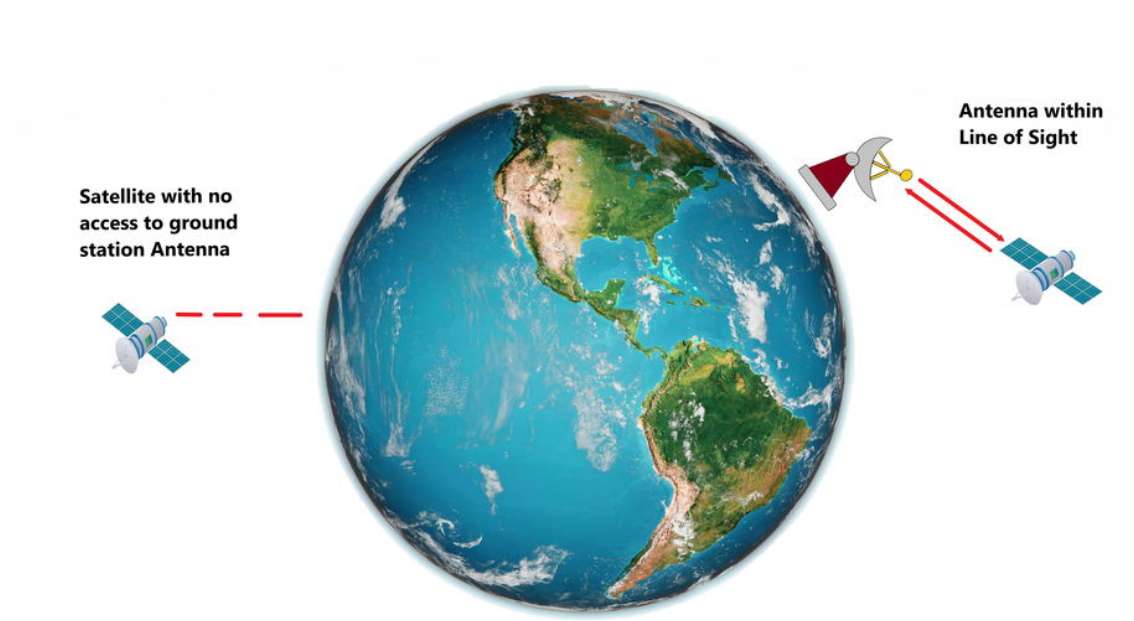

Figure 1.1: Line-of-sight from ground station to a satellite

Edge computing is a promising technology as it involves processing the data close to the source where the data is generated rather than a central server or cloud computing facility. The rise of space data generated and the need for real-time processing of space applications has increased the demand for edge computing in space. It is predicted that until the year 2028, most of the data processing and analysis will take place at the edge of the network [7]. Edge computing is expected to reduce limitations of bandwidth due to the large volume of space data, lower cost due to reduced operation of cloud services and produce real-time results [8]. This technology is currently adopted in several industries such as the medical industry, manufacturing and automotive sectors. It benefits from powerful computation resources and advanced software algorithms [9]. The novel edge learning algorithms can learn from the local data generated in space and extract useful information such as detection of faults in critical parameters. This has also led to the rise of companies investing in edge computing solutions for in-space operations such as OrbitsEdge, Loft Orbital, Orbital Transports etc.

For future space missions to the moon and beyond, network latency will be a critical issue for reliable data transmission to earth. Hence, edge computing solutions like the containerized code solution developed by IBM for the DNA sequencing project on the ISS are useful [10]. There is also an increase in COTS hardware components in space applications which help increase the computing performance for edge computing applications in comparison to rad-hard components which are heavier, provide low performance and have higher costs. The research conducted by Kothari et al. [11] and Furano et al. [12] proposes to run AI algorithms in space to perform inference using COTS hardware accelerators since it shows improved energy efficiency and low costs and mass.

Edge AI is the deployment of artificial intelligence applications close to the source of data using edge computing capabilities. The processing of AI algorithms close to the source rather than the cloud offers faster response times, lower bandwidth costs and re-

silience to network failure. It is important to note that cloud computing can support edge AI deployment during training of the model, processing complex tasks from the edge and storage of data that comes from the edge [13]. This technology is explored further in this thesis.

1.2 Thesis goals and contribution

This thesis proposes the use of edge computing technology to detect faults in space electronics hardware during on-orbit operation which helps in early fault detection and diagnosis. The work focuses on the detection of thermal anomalies in space hardware.

The main goal is to implement an Edge AI application on a compact infrared payload for detecting thermal anomalies on space electronics hardware. The implementation is done using infrared thermography and artificial intelligence techniques. The work aims to deploy and test the Edge AI software on the onboard computer of the payload that will provide edge computing capabilities.

The second goal is to contribute to the development of the compact infrared payload using fast development approach. The thesis contributes to the system modelling of payload, development of AI software, testing of software architecture and assembly of the flight payload. The payload development is inspired by Agile systems engineering methodology to reduce the development time. The development and testing are conducted in the Cubesat of the University of Luxembourg.

Thus, this thesis expands the application of edge computing in space and contributes to the space mission of the University of Luxembourg. It also validates the feasibility of thermal anomaly detection in space despite limitations in storage resources and computing capabilities.

1.3 Thesis structure

The thesis highlights the use of edge computing for space applications. The development of an AI-driven payload involves the design of hardware architecture, software architecture, assembly, integration and testing of the payload.

This document begins with an introduction to the AI4Space mission of the University of Luxembourg in Chapter 2. A summary of the SpaSys research group activities is mentioned. It is followed by the discussion of different deep learning architectures that can support thermal anomaly detection in space electronics.

In Chapter 3, the AI-driven payload is modelled using MBSE techniques. The different components of designing the payload such as hardware and software architectures, testing the mission objectives of the payload and assembling the payload are discussed in detail. Due to the delay in the arrival of hardware components, the testing of only the breadboard model of the payload is discussed. The limitations in the development of

payload are also addressed in this chapter.

Chapter 4 presents the conclusions of the thesis work conducted in the CubeSat lab. The results of the Edge AI software are presented along with the major challenges in the development of the software.

Sec. 4.2 discusses the future development of payload using Autoencoder architectures for anomaly detection. A new hardware configuration using a Xilinx Zync Ultrascale+ MPSoC chip as the onboard computer is discussed that can support edge computing applications in space.

CHAPTER 2

EDGE COMPUTING FOR THERMAL ANOMALY DETECTION IN SPACE

Space industrialization and the development of reusable rocket technology have reduced the cost of launching satellites into space. This has led to the emergence of NewSpace companies which allows launching payloads at cheaper costs and reduced launch schedules. Therefore, it increases the feasibility of testing edge computing capabilities in space.

This thesis focuses on the AI4Space mission of the University of Luxembourg which involves the development of a compact infrared payload using the payload hosting capabilities of a satellite provider, namely Skykraft. The payload aims to detect thermal anomalies in space electronics hardware. A thermal anomaly is defined to be any temperature variation on the thermal image which is outside the nominal temperature range. However, detection of these anomalies in space involves consideration of the space environment, hardware resources and software algorithms.

The thesis uses deep learning, a popular artificial intelligence technique that mimics the human brain [14]. It is a widely used methodology to identify complex features in images and it offers high performance in object detection and image classification tasks. Thus, several deep learning architectures are studied to find the thermal anomalies in the space electronics hardware using the thermal images generated by infrared thermography. Infrared thermography is chosen as a fault detection method for electronics hardware since it is a commonly used non-invasive method to detect thermal anomalies on electronics hardware. This method provides thermal images with temperature variation of electronic components on the electronic hardware by detecting the infrared energy emitted by the components.

The work also uses COTS components to run AI algorithms in space due to their accessibility, low cost and mass. It is important to note that these components are subjected to Total Ionizing Doze and Single Event Effects such as Single Event Latch-ups and Single Event Upsets. It is possible to do polymer or lead shielding to reduce these radiation effects on hardware [5].

The following sections discuss the state of the art of application of edge computing in space missions conducted by the University of Luxembourg.

2.1 Context

The thesis is conducted in collaboration with the SpaSys research group at the University of Luxembourg. The core competency of the group[1] is systems engineering methods focused on miniaturized space systems, service-based space systems design and in-space manufacturing. The mission of the group is to develop space systems and missions via new system engineering approaches and contribute to the sustainable use of space. The research group is also known to develop payloads and chipsats to be sent to space at a higher frequency and short development time.

The CubeSat Lab (shown in Fig. 2.1) in the University of Luxembourg is used for the design of software and hardware architecture, assembly, integration and testing of space systems. It also contains a vacuum chamber and airflow chamber which are used for the development of the space systems. The vacuum chamber shown in Fig. 2.2 is used to simulate the space environment with low-pressure conditions. The airflow chamber or laminar flow cabinet shown in Fig. 2.3 is used for assembling the hardware components and it is used to prevent contamination of the electronic components.

Figure 2.1: Cubesat Lab in the University of Luxembourg

Figure 2.2: Vacuum chamber

Figure 2.3: Airflow chamber

[1] https://wwwfr.uni.lu/snt/research/spasys/research_topics

2.2 The AI4Space mission

The AI4Space mission is a space mission by the University of Luxembourg. The mission is a collaboration between two research groups namely, CVI2 (Computer Vision, Imaging and Machine Intelligence Research Group) and SpaSys (Space Systems Group). The first objective of the space mission is to launch computer vision algorithms into space and the second objective is to demonstrate how space-systems can be built end-to-end using a fast development approach.

The first flight mission involves the development of a compact-infrared payload. This payload aims to detect thermal anomalies on space electronics hardware using Infrared Thermography and AI algorithms. This is also considered a use-case for testing the edge computing application for space. This thesis contributes to the first flight mission of the payload and also concludes with the preliminary development of the second flight mission of the infrared payload.

The payload is launched using a payload hosting service provider, namely Skykraft, an Australian space services company, through their Skyride program. Fig. 2.4 shows a model of the Skykraft spacecraft that hosts the payload. Skykraft plans to host the payload for a period of 90 days in a sun-synchronous orbit. The payload is a circuit board also called "SkyKard" (as shown in Fig. 2.5) with dimensions of 100 mm × 100 mm × 15 mm. The SkyKard receives 5W (5V @1A) power and data via a PCIe connector. The planned launch date for the mission is May 2023.

Figure 2.4: SkyKraft payload hosting service [1]

The results of the space mission are the infrared images and a text file containing thermal anomaly detection information which are received on earth through telemetry. The functionality of the space mission is tested using a breadboard model on-ground before the manufacturing of the SkyKard. The various subsystems, hardware and software architecture will be discussed further in Chapter 3.

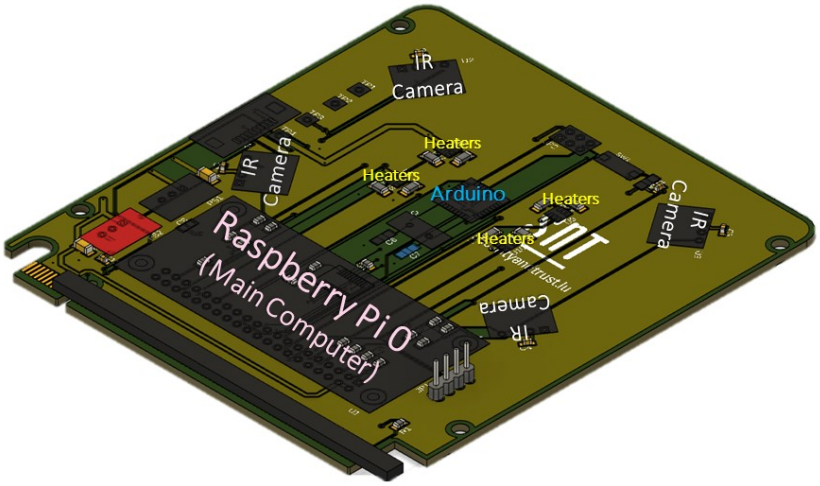

Figure 2.5: SkyKard developed by University of Luxembourg

2.3 Deep learning approach for thermal anomaly detection

This section discusses the study conducted for detecting thermal anomalies on space electronics hardware. For any application, it is important to understand the type of data that is being analysed to detect thermal anomalies. The compact-infrared payload uses Infrared Thermography which provides thermal images captured using infrared cameras. This type of data is unstructured data since it does not have a predefined data model and it is qualitative data (data which cannot be quantified or measured). Thus, this type of data requires advanced algorithms such as deep learning or reinforcement learning algorithms to process them.

It is possible to obtain a global average temperature range of each thermal image generated using Infrared Thermography. This can be classified as structured data and it can be analysed using machine learning algorithms which predict future behaviour based on past data. However, the thesis does not consider this approach since a global temperature range does not take into account all the temperature information on the thermal image. The more information or data, the better the accuracy of detecting thermal anomalies and considering global averages can lead to the loss of information with regards to detecting anomalies. The temporal variation of the image is also not considered in the thesis.

Following the decision to detect thermal anomalies using the captured thermal images of the space electronics hardware, it is important to define what a thermal anomaly is, for our payload. The thesis considers a thermal anomaly to be any temperature variation on the image which is outside the nominal temperature range. The nominal temperature range is not quantified but different simulation scenarios of nominal operation of target hardware on the payload are discussed in chapter 3. Thermal anomalies are usually

detected by comparing the thermal image with a reference image. However, the space environment is harsh and the thermal anomalies vary with the orbital conditions/situations. The advanced AI algorithms can detect the temperature variations of each pixel value of the thermal image by learning from the space environment and this is useful to detect thermal anomalies. This helps in pixel-wise comparison between the learned reference temperatures with the actual temperature values.

The deep learning algorithms are a class of AI algorithms which are gaining popularity due to high performance for image classification and object detection applications [14]. Deep learning is also considered a promising technology for fault diagnosis on spacecraft since it can automatically acquire diagnostic features and reduces the burden of manual fault feature extraction [15] [16]. The increased performance can be attributed to an increase in data, training techniques and computing performance of hardware. These algorithms mimic the human brain and use concepts of artificial neural networks for prediction. A deep learning architecture consists of an input layer, hidden layers and an output layer as shown in Fig. 2.6. The input layer can be pixels of an image, parameters causing specific medical diseases etc. The hidden layers build a complex mathematical relationship between the input neurons of the input layer and the output neurons of the output layer. They are used to learn features of an input image or capture mathematical representation between input and output parameters. The output layer consists of output neurons which are probability values of input parameters belonging to certain classes (ex: anomaly or not anomaly, dogs or cats, cancer, diabetes or fever etc).

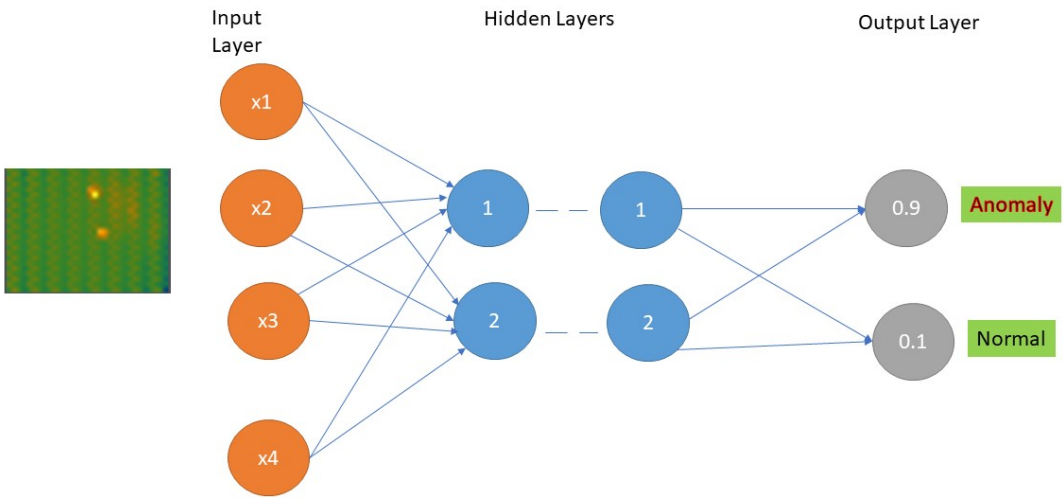

Figure 2.6: A sample deep learning architecture

In this thesis, a study was conducted on different deep learning algorithms that can be used for thermal anomaly detection for space applications. Some popular deep learning algorithms include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Restricted Boltzmann Machines (RBMs) and Autoencoders [17]. The deep learn-

ing architecture is classified into supervised, semi-supervised and unsupervised learning algorithms. Supervised learning for anomaly detection will involve training the algorithm with a labelled dataset consisting of normal and abnormal (shows anomalous behaviour) images. This methodology will involve manual segregation of the dataset. The unsupervised learning involves training the algorithm with no labelled dataset and hence it avoids manual work [18]. The algorithm has to find patterns with the input data and perform segregation of inputs into the normal or abnormal image. The semi-supervised learning uses both labelled and unlabelled data for training the algorithm [19].

It is to be noted that thermal images have low contrast, lower resolution and lower signal-to-noise ratio than optical images which makes segmentation of different objects (i.e. resistor, capacitor, on-board-computer etc.) and identifying regions of interest difficult in these images. The research conducted by Cai Lile et al. [20] proposes to use input visible images to predict thermal images and then compare these predictions with the ground truth images for training the CNN. Similarly, the paper by Gwanyong Park et al. [21] uses thermal and visible images for thermal anomaly detection on walls by using CNN for visible image segmentation and then detects anomalies through temperature distribution of the thermal images. However, these methodologies cannot be adapted for the space payload since it is difficult to obtain optical images inside the satellite due to lack of ambient light and also because of the space environment which gives rise to shadows on the satellite.

This thesis has studied the supervised and unsupervised learning algorithms that are useful for thermal anomaly detection on thermal images. The algorithms suitable for edge computing applications are chosen due to the high computation requirement of deep learning algorithms.

For supervised learning, Pérez-Aguilar et al. [22] observed that the deep learning architectures that were previously trained had higher accuracy and hence, transfer learning through the use of previously trained supervised learning architectures is considered suitable for thermal image classification. The two popular supervised learning architectures in studied in sec. 2.3.1 and sec. 2.3.2.

2.3.1 MobileNetV2 Architecture

MobileNet-v2 is interesting deep learning architecture for the thermal anomaly detection application since it is consuming low memory (13.6 kB) and having low training parameters (3.5 million) in comparison to other deep learning architectures and it is found to still deliver comparative performance in terms of accuracy of image classification. This algorithm is suitable for binary classification which can classify the captured thermal images as normal or anomalous images.

MobileNet architectures are built to be suitable for mobile applications and resource-constrained environments [23]. The MobileNetV2 architecture is an improvement over the MobileNetV1 architecture and uses depthwise separable convolutions to filter features, linear bottlenecks to extract manifolds of interest and inverted residuals that use an inverted structure for efficiency. The MobileNetV2 models use 3.4M parameters with 300M Multiply-Adds for ImageNet Classification [24]. Fig. 2.7 shows the architecture

can be used for classification, object detection and segmentation tasks and it is found to be the most efficient model in comparison to prior state of art models (i.e. ShuffleNet, YOLOv2, ResNet) with fewer training parameters and less number of Multiply-Adds. Thus, this is a suitable architecture for thermal anomaly detection in embedded devices.

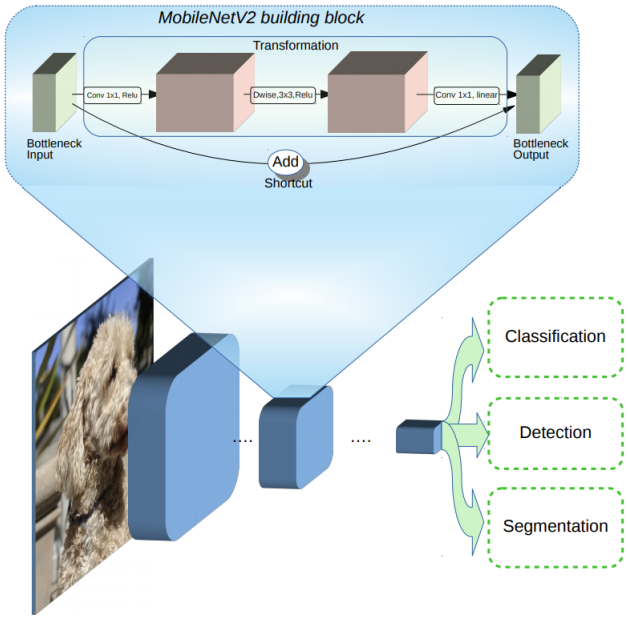

Figure 2.7: MobileNetV2 architecture [2]

2.3.2 EfficientDet Architecture

The use of AI techniques for thermal anomaly detection in different applications was studied to understand if it can be adapted to the space payload in this thesis. The research by G. Cipriani et al. [25] uses EfficientDet, an open-source CNN architecture (supervised learning algorithm), to achieve up to 99% accuracy for the automatic inspection of thermographic images to detect anomalies in photovoltaic plants. This methodology also reduced the presence of human error and reduced the time for a system inspection. The same research approach with EfficientDet architecture was used for thermal anomaly detection in induction motors and accuracy of up to 100% was achieved [26]. It is to be noted that the accuracy depended on the location of the thermal anomaly on the motor and detection of anomalies took up to 34 seconds.

EfficientDet as shown in Fig. 2.8 is an object detection architecture developed by Google Research, Brain Team. This architecture can help identify the parts (i.e location) in the image with anomalous behaviour in addition to classifying the image as anomalous. The EfficientDet architecture proposes several optimizations to existing object detection architecture to improve efficiency such as a weighted bi-directional feature pyramid network to enable multi-scale feature fusion and a compound scaling method that uniformly scales the depth, width and resolution of neural network [27]. The architecture was built to develop efficient detector architecture that can fit into real-world applications containing embedded devices with resource constraints (memory, computation power etc.), for

ex. EfficientDet-D0 architecture has 3.9M parameters and 2.5B FLOPs. It achieves better accuracy and efficiency with fewer FLOPs than popular object detection architectures such as YOLOv3, RetinaNet and ResNet architectures. Thus, this architecture is a suitable object detection algorithm for use in devices with resource constraints and also the thesis application of compact infrared payload.

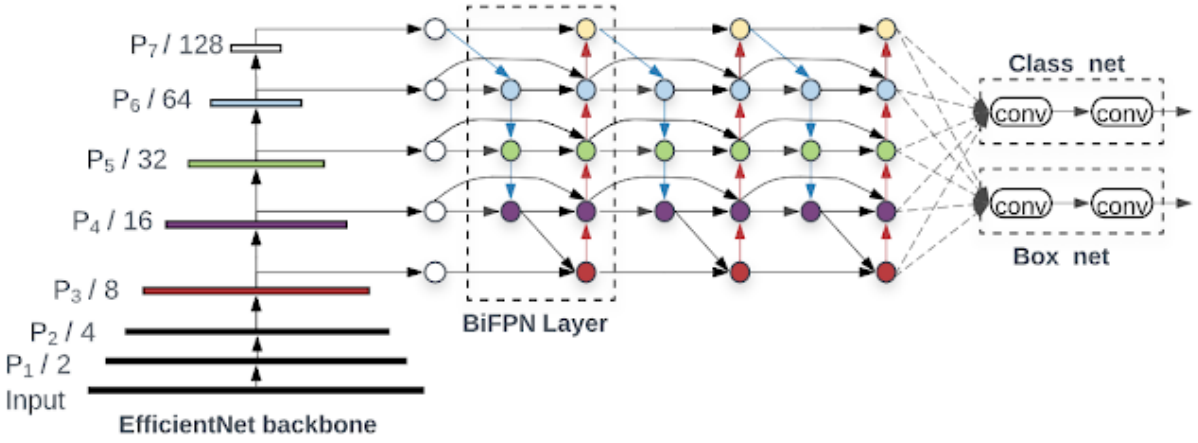

Figure 2.8: EfficientDet architecture [3]

2.3.3 Unsupervised Learning

Unsupervised learning seems promising due to the elimination of image segmentation and the need for a labelled dataset for training. Unsupervised learning infers a structure out of input data points and provides its definition of anomalous data. Autoencoders belong to a class of deep learning algorithms for unsupervised learning. A poor autoencoder can be a good anomaly detector if there is a strong bias that favours the reconstruction of background data whereas a good autoencoder can also reconstruct anomalies. This was observed in the use of autoencoders for unsupervised anomaly detection in top jet images [28]. The research conducted by Christoph Baur et al. [29] uses autoencoders to detect anomalies in Brain MR Images. The methodology used by the research uses autoencoders to learn a model of normal anatomy and then recover healthy data. While anomalous images generate erroneous data when passed through the same model. However, the models performed poorly in the reconstruction of healthy parts of the anomalous samples. Unsupervised learning is a potential research direction with its challenges of building an accurate model for detecting anomalies in images. These algorithms are not explored further in this thesis and included in the future work, sec. 4.2.

2.3.4 Conclusion of study

Deep learning algorithms were found to be suitable for thermal anomaly detection and transfer learning was found to give better performance on deep learning architectures. Then, a further study was conducted on some of the popular supervised and unsupervised learning algorithms in deep learning for deployment on embedded devices. Fig.

2.9 shows the different algorithms explored under supervised and unsupervised learning algorithms, namely MobileNetV2, EfficientDet and Autoencoders.

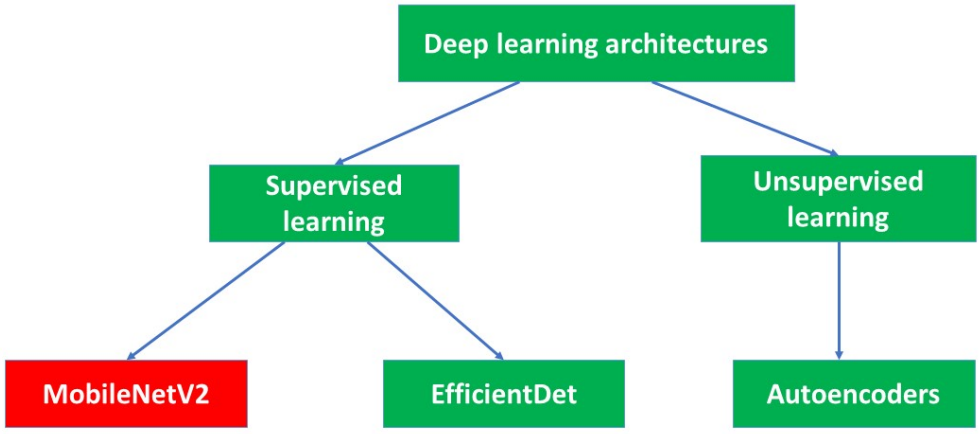

Figure 2.9: MobileNetV2 architecture chosen for thermal anomaly detection

The research concludes that the MobileNetV2 algorithm will be a good starting point for binary classification of thermal images for detecting thermal anomalies on the compact-infrared payload. The reasons are the following:

1. They have a small architecture which leads to fewer computations/operations, low training parameters and consumes low memory
2. Low computation time due to small architecture
3. Elimination of making bounding boxes for training the thermal images in comparison to EfficientDet architecture. Preparation of labelled bounding boxes on thermal images is difficult due to the low contrast of thermal images
4. output also provides the probability of the thermal image is an anomaly
5. It is widely used in many embedded applications

The unsupervised learning algorithms are potential algorithms for thermal anomaly detection. However, they require further research studies to be conducted for deployment on embedded devices.

CHAPTER 3

DEVELOPMENT OF AI-DRIVEN PAYLOAD

This chapter discusses the development of an Edge-AI application for space i.e. compact-infrared payload. It begins with system modelling of the compact infrared payload using MBSE methodology. Later, my contribution to the different subsystems of the payload is highlighted including the explanation of hardware architecture, software architecture, assembly and testing of the payload. In addition, the AI technique chosen in the previous chapter is implemented on the first configuration of the compact infrared payload with Raspberry pi as the onboard computer.

The steps for space system (i.e. payload) development in the SpaSys research group is indicated in Fig. 3.1. The space mission begins with understanding the mission needs and objectives [30] [31]. The development is also inspired by Agile Systems Engineering [32]. Hence, the requirements diagram and system modelling are not done at the start of the development. This is followed by the space system design wherein the different components, interfaces, modules and data are defined to meet the mission objectives. Then, the system development includes the design of software and hardware architectures. After the development, the different components of the payload are assembled, integrated and tested in the CubeSat Lab. The payload will be qualified for launch only after successful verification and validation of the functionalities and this is important for it to operate in space. This thesis does not discuss the launch and operation, since the launch is scheduled for early 2023.

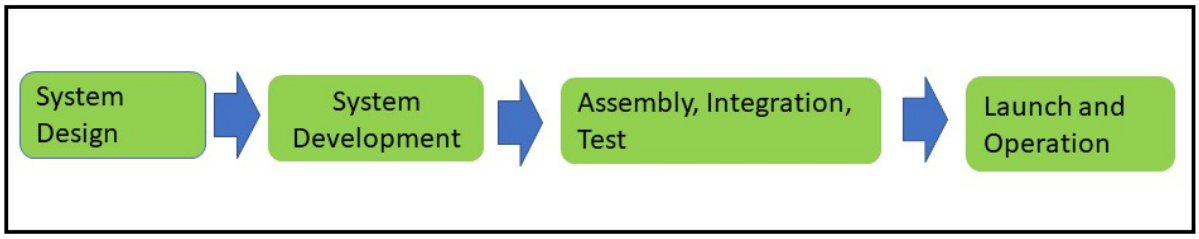

Figure 3.1: Space systems development

3.1 Software tools used

To fulfil the objectives of the development of the AI-driven payload and tests its operation, several tools are used and this is illustrated in Fig. 3.2. The following tools are crucial for the development process:

1. Ttool [1] : A toolkit to create SysML models
2. GitLab [2]: An application for software management and sharing source code within the team
3. Edge Impulse [3] : A development platform for machine learning on edge devices
4. KiCad [4]: The electronic design suite used to visualize the hardware architecture of the payload
5. Visual Studio Code [5]: A code editor used in thesis

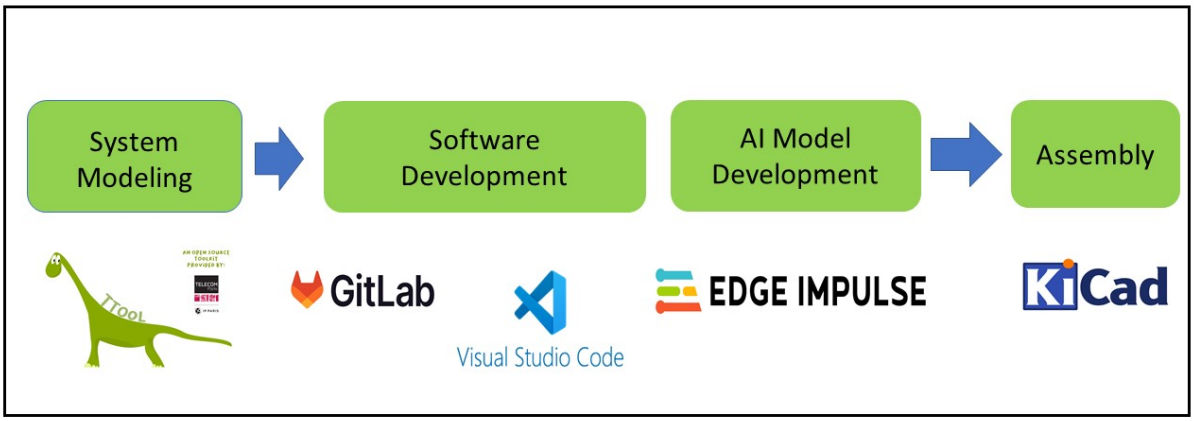

Figure 3.2: Software tools used in the thesis

[1] https://ttool.telecom-paris.fr/
[2] https://gitlab.uni.lu/
[3] https://www.edgeimpulse.com/
[4] https://www.kicad.org/
[5] https://code.visualstudio.com/

3.2 MBSE modeling of payload

According to INCOSE [6], "Model-based systems engineering (MBSE) is the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [33]. MBSE allows system development with the use of models and moves from document-centric to model-centric. This model-centric approach is currently used in the development of complex systems such as Aerospace, Automotive Manufacturing etc.

In this thesis, SysML models are used to understand the compact infrared payload as a whole with its different sub-systems along with its interactions, requirements, behaviours, properties and interconnections. These models act as a reference when several engineers are working on the same system and reduce the complexity of understanding the system. Ttool is the software used to build these models and the models are inspired by the MBSE course in ISAE-SUPAERO.

The 4 pillars of SysML include the Structure Diagram, the Behavior Diagram, the Requirements Diagram and the Parametric Diagram. The system is defined to be a thermal anomaly detector or the computer module which contains the embedded software for thermal anomaly detection. Some of these diagrams are modelled for understanding the compact infrared payload as indicated below: [33]

1. Needs: Before starting the SysML modelling, it is important to understand the payload expectations, objectives, issues, perceived risks and opportunities from the stakeholder's perspective. This is indicated in Fig. 3.3. Some of the needs include ensuring that the temperature of target electronics is within the nominal temperature conditions required for its operation and detecting the thermal anomalies from the captured thermal images.

2. Assumptions: After understanding the needs, the modelling assumptions were noted. Fig. 3.4 indicates the assumptions that were considered when modelling the system (i.e. thermal anomaly detector). This includes the assumption that the satellite payload is a simplified and perfect system. The failure of the payload due to the space environment or insufficient power is not modelled.

3. Requirements diagram: Requirements define what is required from a system (i.e. thermal anomaly detector) to be built. The diagram (shown in Fig. 3.5) contains the functional requirements that specify the functions that the system must perform during space flight (ex: The system shall capture thermal images of electronics from each IR camera) and non-functional requirements help measure the effectiveness of the system (ex: The system must take a picture every 5-6 minutes). It provides the specifications to start the development of the system. Many parameters of the payload were not fixed during the initial phase of development and hence some specifications are roughly estimated and indicated as TBD (To Be Discussed).

[6] https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf

4. Use case diagram: This diagram in Fig. 3.6 is part of the behaviour diagram in SysML. The use case diagram identifies the main functions and services to be offered by the system (i.e. thermal anomaly detector). It also considers the interactions between the system and the actors (role of users) it interacts with (environment). The main system is a thermal anomaly detector which is the computer module or Raspberry Pi and it runs the onboard software of the satellite payload. The primary actor is the Mothercraft which directly interacts with the system and receives thermal images from the system. The secondary actors are triggered by the use-case such as IR Camera Driver, Arduino, Heaters and Compact Flash Driver.

5. Sequence diagram: This diagram is part of the behaviour diagram in SysML. It models the interactions between actors and the system in a time sequence. Fig. 3.7 demonstrates the sequence in which different actors work together when an anomaly is detected by the thermal anomaly detector (system). This sequence starts with the system changing the clock speed of Arduino and activating the heaters to simulate anomalous behaviour. Then, the system detects the anomaly in the captured thermal image after running the AI program on the thermal image. Later, it sends the thermal image with anomaly to Mothercraft. A similar sequence is modelled in Fig. 3.8 which demonstrates the sequence in which different actors communicate with each other during the normal operation of the payload. The sequence diagram also shows the messages transmitted between the system and the actors such as activate heaters, change clock speed etc.

6. Internal Block diagram: It is part of the Structure diagram in SysML. It models the high-level architecture in terms of communicating blocks and the different inputs and output methods of each block. The thermal anomaly detector (computer module) is the subsystem in the focus of the payload. The input and output signals through the ports of each block are also indicated in Fig. 3.9. The different interacting blocks include Thermal anomaly detector, Arduino, Mothercraft, IR Camera Driver, Heaters and Compact Flash Driver.

It is observed that the mission needs and the internal block diagram add maximum value to understanding the compact infrared payload (system) as a whole. The fast development approach adopted in the development of the payload demands project execution in less time. Hence, it is not necessary to know the requirements for building agile systems as these requirements can be modified during the project development. The Use case diagram and Sequence diagram are useful in understanding the communication between different actors and the environment, however, they add overhead to the development time and can be eliminated for miniaturised spacecraft system design.

After understanding the system architecture using SysML models. The hardware and software architecture required to implement the thermal anomaly detector algorithm on the payload is studied in detail in the next sections.

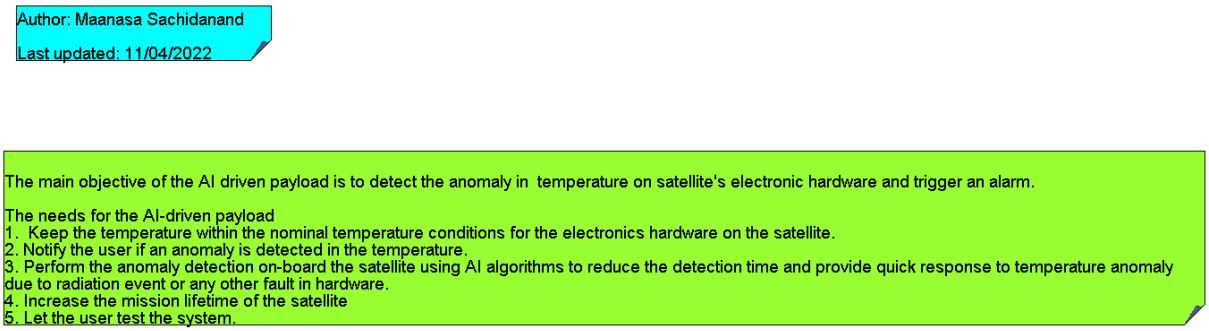

Figure 3.3: Mission Needs and Objectives

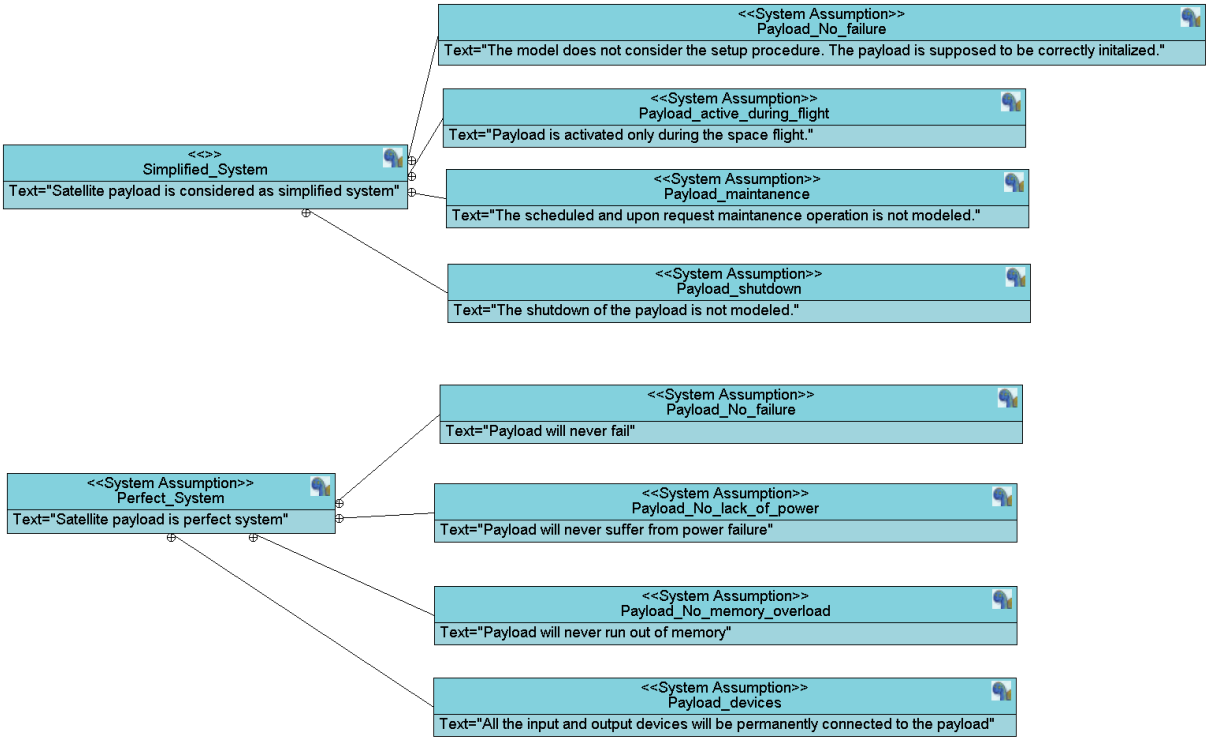

Figure 3.4: Modeling Assumptions

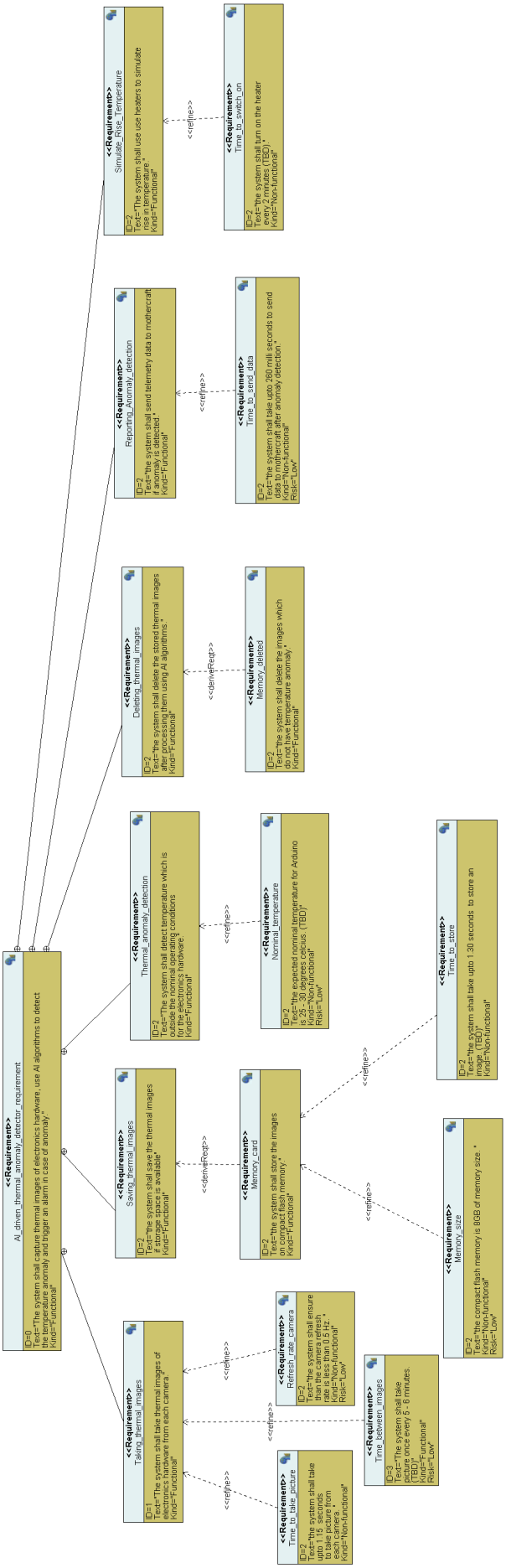

Figure 3.5: Requirements Diagram

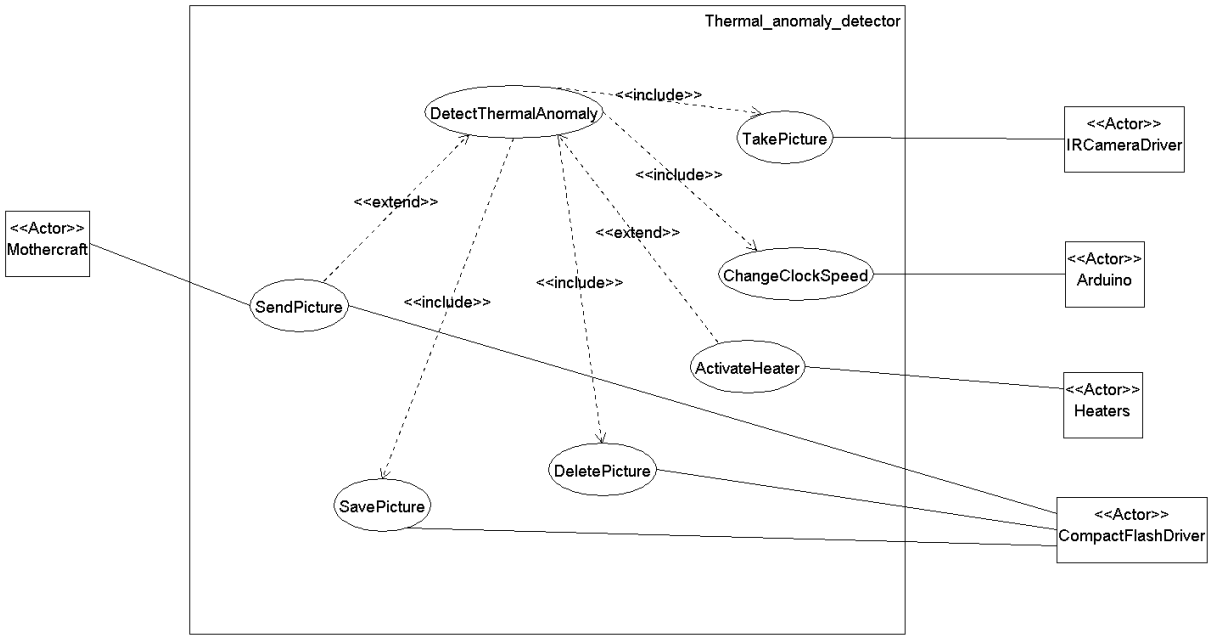

Figure 3.6: Use case diagram

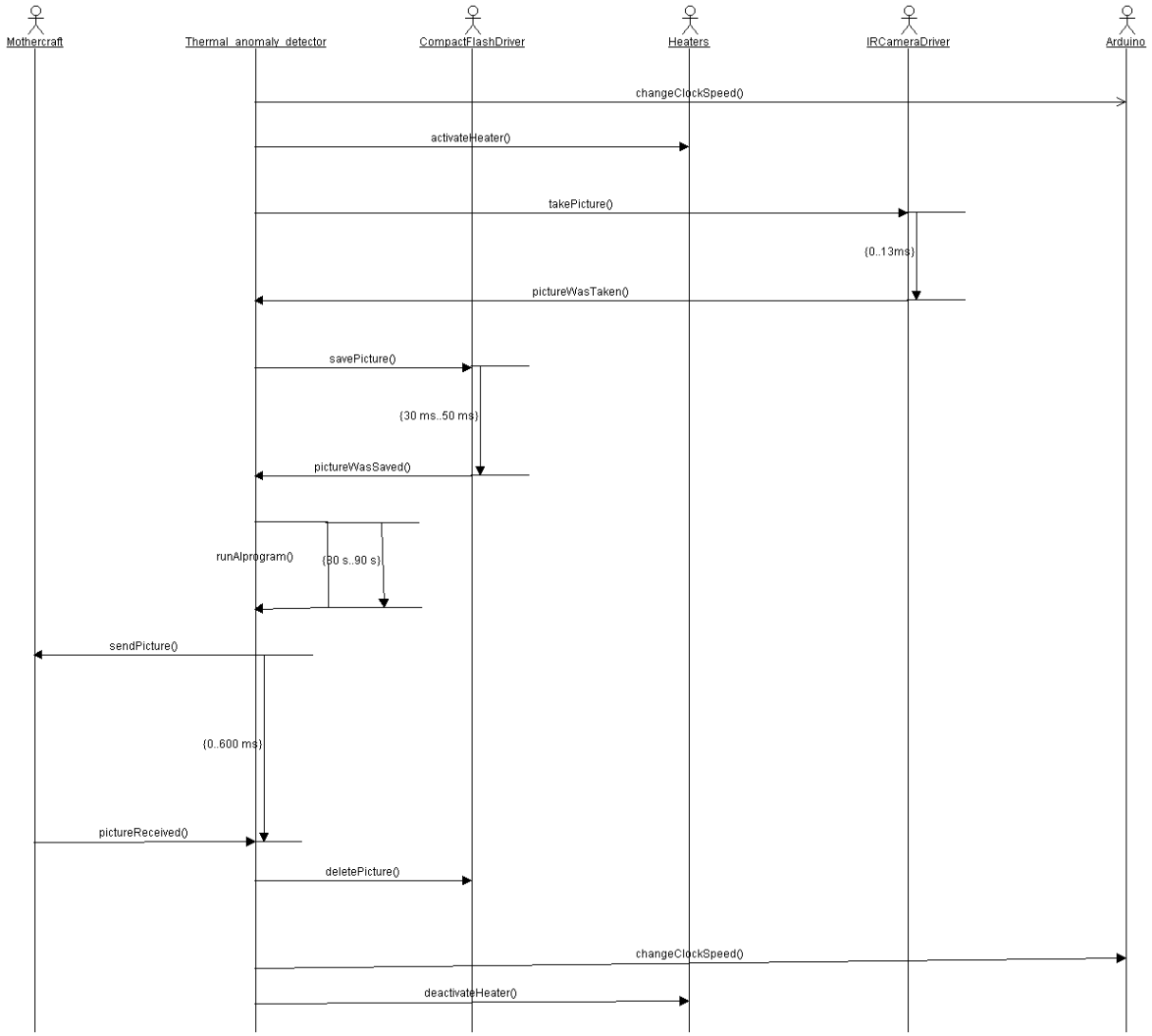

Figure 3.7: Sequence diagram: Scenario with thermal anomaly

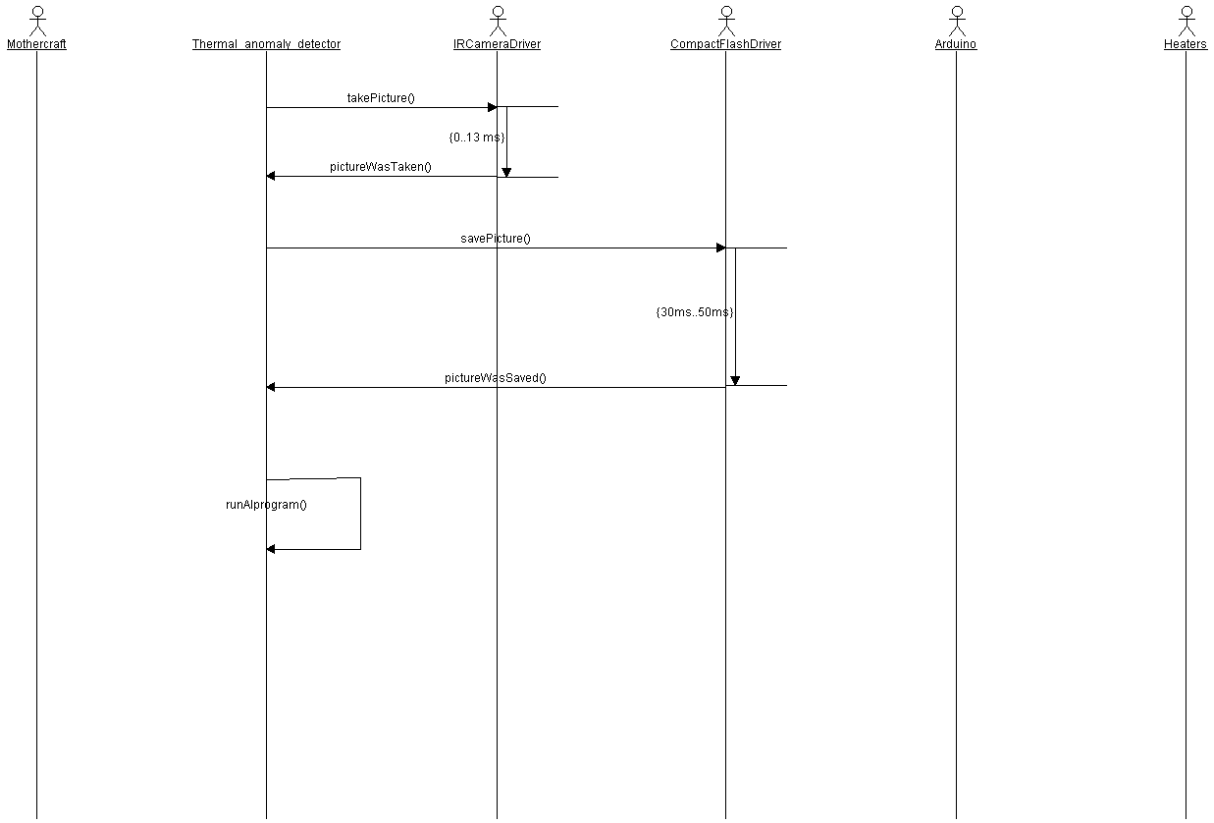

Figure 3.8: Sequence diagram: Scenario without thermal anomaly

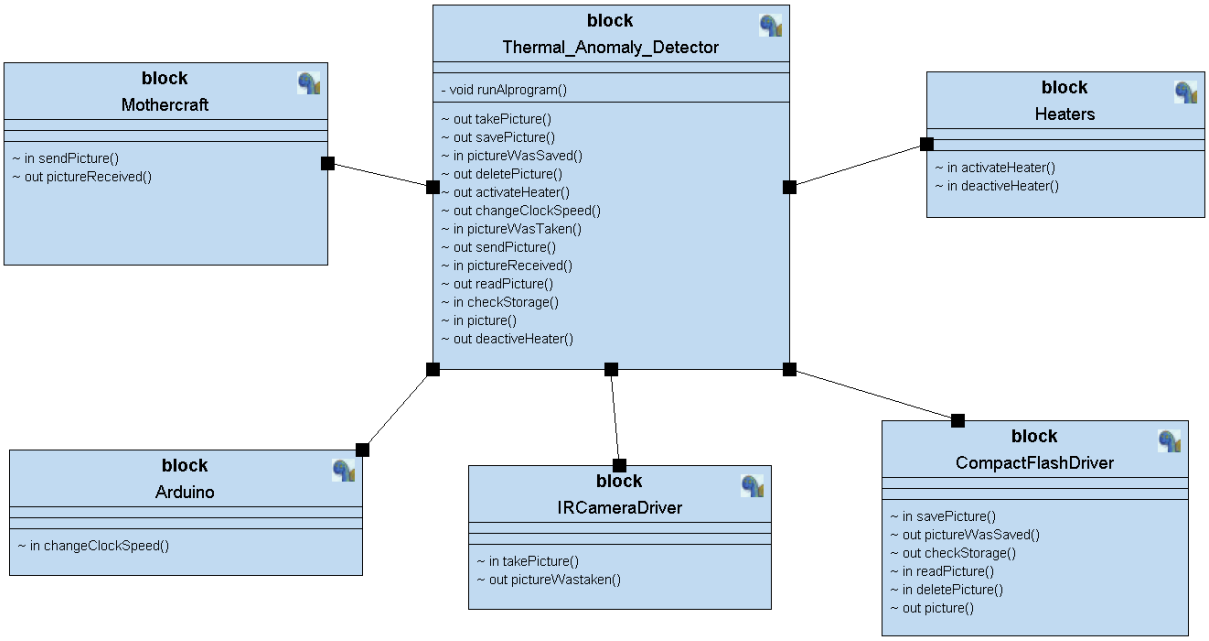

Figure 3.9: Internal Block Diagram

3.3 Hardware architecture of payload

The following section mentions the key hardware components of the first configuration of the compact infrared payload. The hardware architecture (shown in Fig. 3.10) is an important consideration for edge computing applications and the implementation of the AI model. One can also observe the use of COTS components to minimise cost, increase availability of the product and reduce development time.

The subsystems of the payload include:

1. Computer Module - Raspberry Pi Zero 2W is the computer module with specifications of quad-core 64-bit Arm Cortex-A53 CPU clocked at 1GHz and 512MB SDRAM. It requires a 5V supply voltage and has an operating temperature between -20°C to 70°C. [34] The onboard software is embedded in the computer module.

2. IR camera - It is used to capture thermal images of the target electronics on the payload. The MLX90640 camera is a fully calibrated 32x24 pixels thermal IR array with 3.3V supply voltage and operating temperature of -40°C to 85°C. The camera can detect a target temperature between -40°C to 300°C with the precision of 1°C. The outputs of the camera sensor are accessible through I^2C communication. [35]

3. Target electronics - The Arduino (ATMEGA328) and heaters (resistors) are the objects of interest for detecting thermal anomalies. The clock frequency of the Arduino is changed to increase or decrease the temperature and the heaters are turned on using FET bus switch to increase the temperature of the target electronics.

4. Power interface - The payload receives a supply of 5V @ 1A from the mothercraft avionics and the power interface consists of a DC-DC converter to supply 3.3V for Arduino, heaters and other components.

5. Mothercraft interface - The mothercraft hosting the payload communicates with the payload through the backplane board. The spacecraft's onboard computer sends data to the payload using the UART communication protocol. The backplane board of the mothercraft use PCIe connectors to route data and power to the payload. This interface is specific to the spacecraft of the Skyride hosting program of Skykraft.

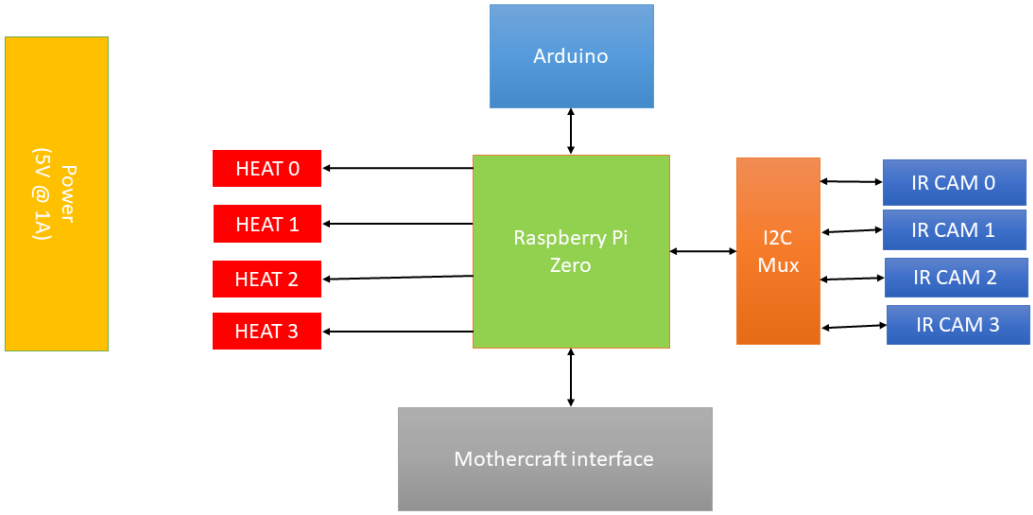

Figure 3.10: Hardware Architecture

3.4 Software architecture of payload

After understanding the different subsystems of the payload, the software architecture developed for the AI4Space mission is studied. Fig. 3.11 shows the AI Detector highlighted as a red block that is integrated into the software architecture. The figure contains two grey blocks, namely the ground segment software embedded on the Raspberry Pi 3 (AI4EGSE) and the flight software called RABITS (Request + Answer-Based Interface for Thermography in Space) embedded on Raspberry Pi Zero (AI4). This thesis focuses on the AI detector software that is developed and integrated into the flight software embedded in the Raspberry Pi Zero. The Python programming language is used to develop the AI Detector to take advantage of some useful Python libraries. The software is built on top of the Raspberry Pi OS installed on the AI4.

The software elements essential for the development of the thermal anomaly detector are summarised below:

1. RabbitMQ[7] - It is an open source message broker which accepts and forwards the messages. It implements the advanced message queuing protocol (AMQP). The messaging model consists of a producer which is a program that sends messages, a queue is a buffer that stores the messages, an exchange transfers messages from the producer to one or many queues and a consumer is a program which receives the messages.

2. AsyncIO[8] - It is a python library which supports concurrent code, i.e managing two or more programs running at the same time.

[7]https://www.rabbitmq.com/
[8]https://docs.python.org/3/library/asyncio.html

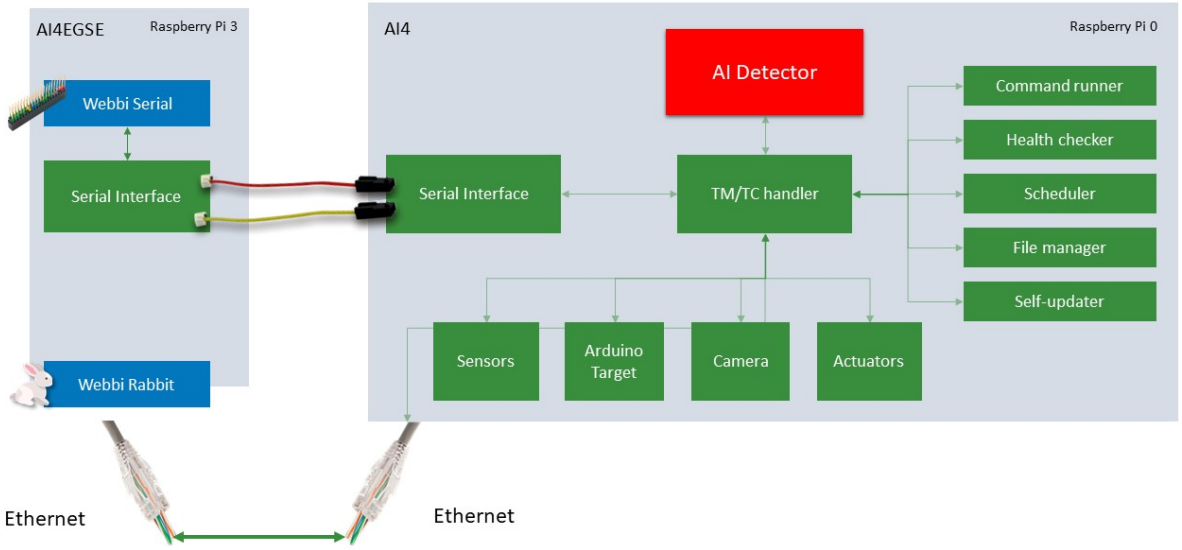

Figure 3.11: AI detector integrated into the software architecture

3. Logging[9] - It is a python library to keep track of events in a program and also report error events when the software runs.

4. Edge Impulse Runner [10] - It runs the AI models built using the Edge Impulse platform on the development board

5. OpenCV-Python[11] - It is a library of Python bindings designed to solve computer vision problems. This library is used to read the captured thermal images stored in PNG format.

6. Supervisor[12] - It is a client/server system that allows monitoring and control processes on the Raspberry Pi OS. It is used to launch, restart and terminate the programs specific to the project that is running on the OS.

The AI detector software runs the AI model developed using the Edge Impulse platform on the Raspberry pi Zero and this is discussed further in sec. 3.5. The software runs every time an image is captured by the IR camera. The output of the AI model (i.e. MobileNetV2 architecture) is the probability of the image is an anomaly and not an anomaly. The output probability values and the image location details are stored in a text file.

The Webbi Rabbit is the web interface that is developed using Javascript programming language on the front end, Python programming language on the back end and Socket.IO library for building real-time web applications. The Webbi Rabbit is implemented on the ground segment hardware (i.e. Raspberry Pi 3) and it helps the user capture the

[9]https://docs.python.org/3/howto/logging.html
[10]https://docs.edgeimpulse.com/docs/edge-impulse-cli/cli-run-impulse
[11]https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
[12]http://supervisord.org/

thermal image, turn on heaters, track anomalies detected on the thermal image and other functionalities. This thesis contributed to the AI detector highlighted in red in Fig. 3.12.

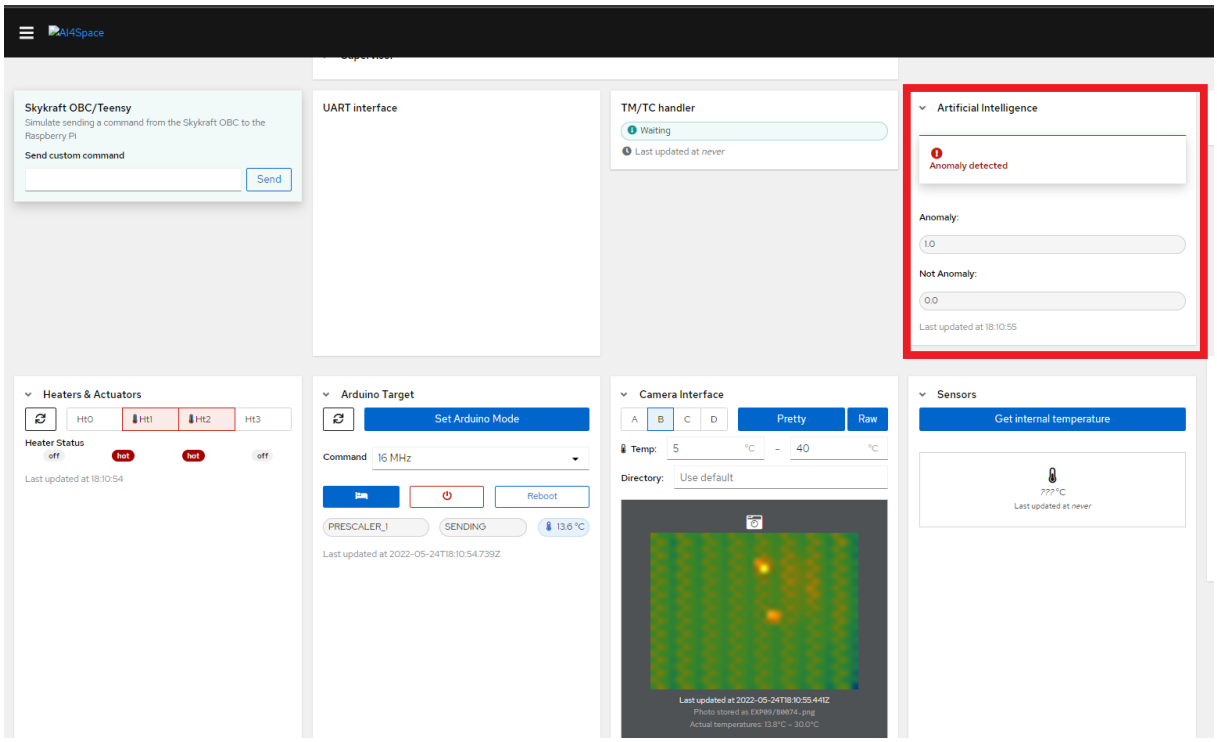

Figure 3.12: AI detector integrated into the web interface

3.5 Implementation of AI detector software

This section explains the implementation of the AI detector software in detail. The software is embedded on the breadboard model of payload shown in Fig. 3.13. It is important to note that the breadboard model of the payload is created to test the functionality of the payload and it does not emulate the flight model of the payload.

The following factors affect the output of the thermal anomaly detection software:

1. Thermal camera placement - The flight model shown in Fig. 2.5 has four thermal cameras which are located at 45° angle to the centre of the Arduino component. It is difficult to ensure that the thermal cameras can maintain this angle on a breadboard model. For the functional testing of the AI detector software, two cameras which have two heaters in their field of view are chosen for testing the breadboard model.

2. Vacuum chamber test - The size of the breadboard model is larger due to the long wires and large size of the hardware components. Due to this reason, vacuum chamber testing could not be carried out using this model. Thus, the pressure changes are not considered for the test. This also restricts the simulation of the space environment for training and testing the AI model.

3. Space environment - The temperature inside the spacecraft vault containing the payloads will affect the performance of the trained AI model. However, this temperature value was not known during the development of the payload and it is ignored for testing the breadboard model of the payload.

   The spacecraft will experience low temperature during the eclipse phase when it is hidden from the sun and high temperature when the spacecraft is closer to the sun. This degree of testing is outside the scope of this thesis. However, the breadboard model was tested during the night and daytime to simulate temperature changes.

4. Vibration test - The payload can experience strong mechanical vibrations during a rocket launch. The payload components specifically the thermal cameras can be damaged due to the mechanical vibrations. This can affect the performance of thermal anomaly detection during spacecraft flight. However, the mechanical vibrations are not considered for testing in this thesis.

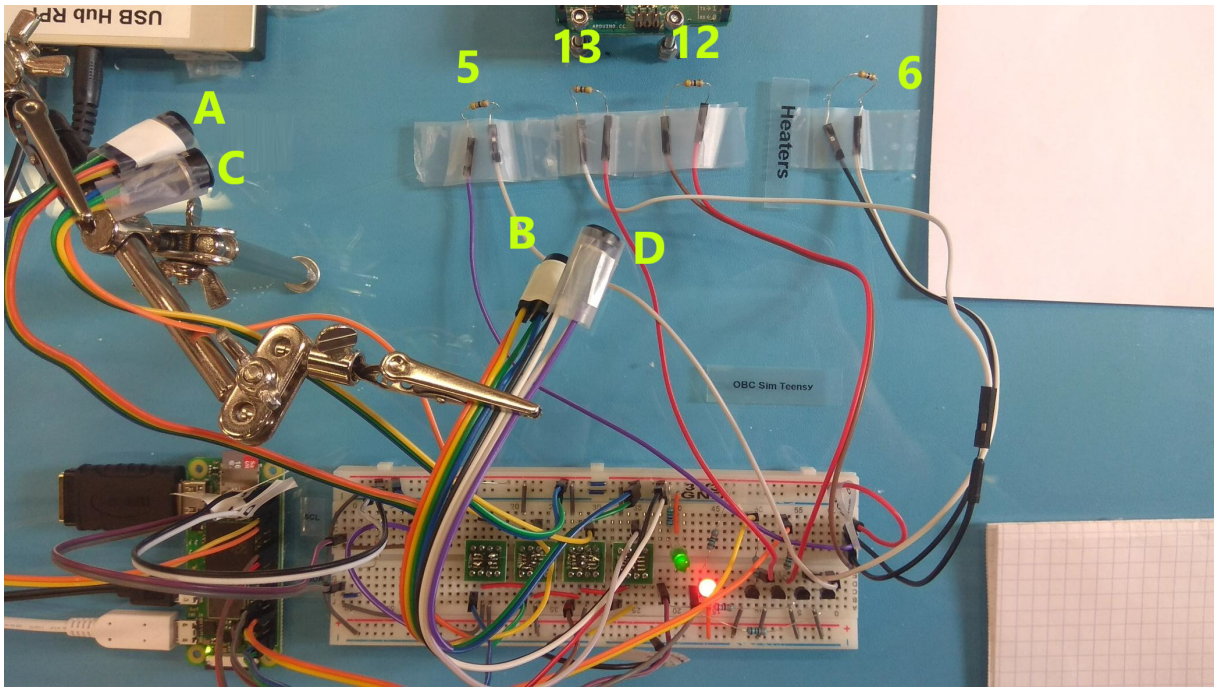

Figure 3.13: The hardware setup for functional test of payload

The four thermal cameras are indicated as A, B, C and D and the four heaters are marked as 5, 13, 12 and 6 in Fig. 3.13. The proof of concept of thermal anomaly detection is done by choosing certain hardware components on the breadboard model of the payload. The components chosen for the functional testing of the AI detection software include:

1. Raspberry Pi Zero (the computer module)

2. The thermal camera marked as D (Note: It can be used interchangeably with thermal camera B due to the camera placement and hence, overlapping Field of View)

3. The heaters marked 13 and 12 (they are in the field of view of cameras B and D)

After fixing the hardware setup for testing the breadboard model of payload, different steps are carried out to build the AI model as indicated in Fig. 3.14.

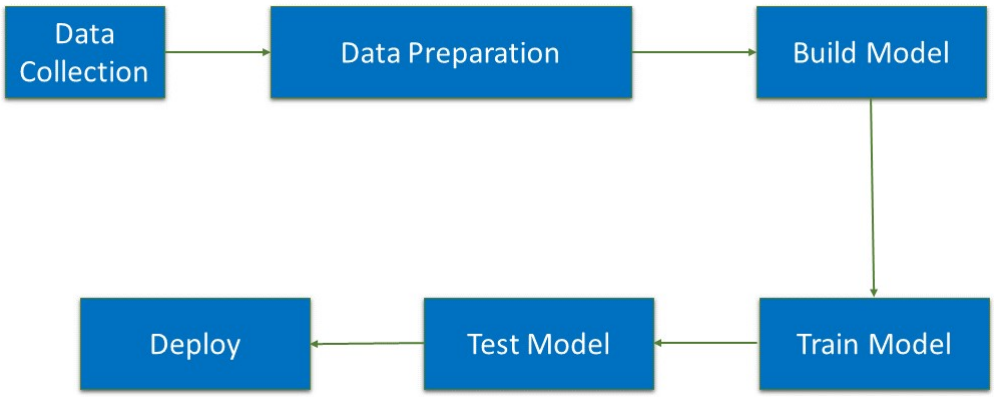

Figure 3.14: Steps to build AI model

Data Collection

The first step to building an AI model is to collect the relevant data. The input data for the AI model is thermal images captured using the IR cameras and this choice is explained in the previous chapter in sec. 2.3.

The original output of IR cameras gives 768 pixel values with each pixel consuming 12 bits but the chosen AI models (ex: MobilenetV2) require RGB images with more pixel values as input data. Hence, the output of these cameras is converted into a heatmap; which is low-contrast RGB images representing temperature variation. Then, these thermal images are magnified 10 times using interpolation techniques for easy visualization and also for use in the AI model. Chapter 2 defines thermal anomalies as any temperature variation on the thermal image which is outside the nominal temperature range. The nominal temperature range is assumed between 20°C to 40°C for functional testing of the breadboard model of payload and this information is required to generate a heat map from the IR camera output. Then, these thermal images are stored as 8-bit PNG image format on the computer module and interpolation techniques are performed as mentioned earlier.

To test the detection of the thermal anomalies, synthetic thermal image data is generated. The synthetic thermal image data are generated using simulation scenarios (ex: turning on the heater to generate an anomalous image) that are not generated by the real-world performance of the hardware. The thermal images captured by the IR cameras using simulation scenarios are classified as follows:

- Normal image - This thermal image is captured when the heaters 13 and 12 are turned off.

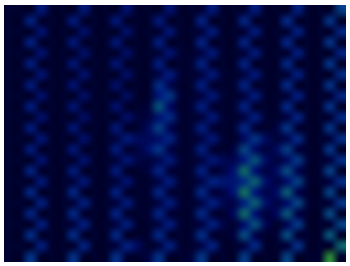

Figure 3.15: Thermal image captured during normal operation of the payload

- Anomalous image - This thermal image is captured when either (shown in Fig. 3.17) or both of the heaters 13 and 12 are turned on (shown in Fig. 3.16).

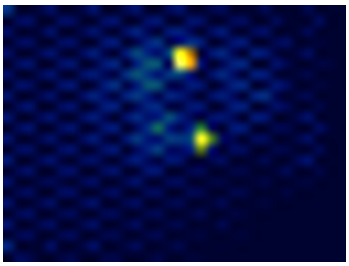

Figure 3.16: Anomalous image: Two heaters are turned on

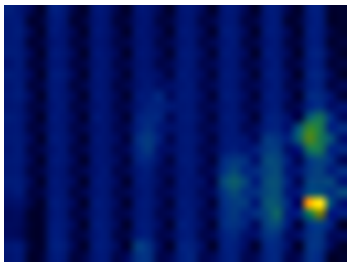

Figure 3.17: Anomalous image: One heater is turned on

About 1050 thermal images (including anomalous and normal images) were captured for 5 days or 120 hours to simulate temperature changes in the environment given that the temperature during the daytime is higher than during the nighttime. Even though anomalous behaviour is a more rare occurrence than normal operation, It is important to have balanced data set (equal representation of each class) for training the AI model to avoid bias in favour of the majority class. In the thermal anomaly detection AI model, we have two classes of output namely Anomaly or Normal (i.e Not-Anomaly). A balanced data set is generated using synthetic thermal image data generation as explained earlier. Hence, the synthetic minority oversampling technique is used to maintain 50% of anomalous images and 50% of normal images in the collected samples.

The Edge Impulse platform is used to build the AI model. Fig. 3.18 shows the creation of a project to detect thermal anomalies in space hardware and 1050 thermal images are uploaded to the platform. The thermal images that are collected are divided into a Training data set and a Test data set. The research [36] suggests best results for the AI model when 80% of the images are used for Training and the rest 20% of the images are used for Testing. Fig. 3.19 shows 863 thermal images are used for training and Fig. 3.20 shows 187 thermal images are used for testing the AI model.

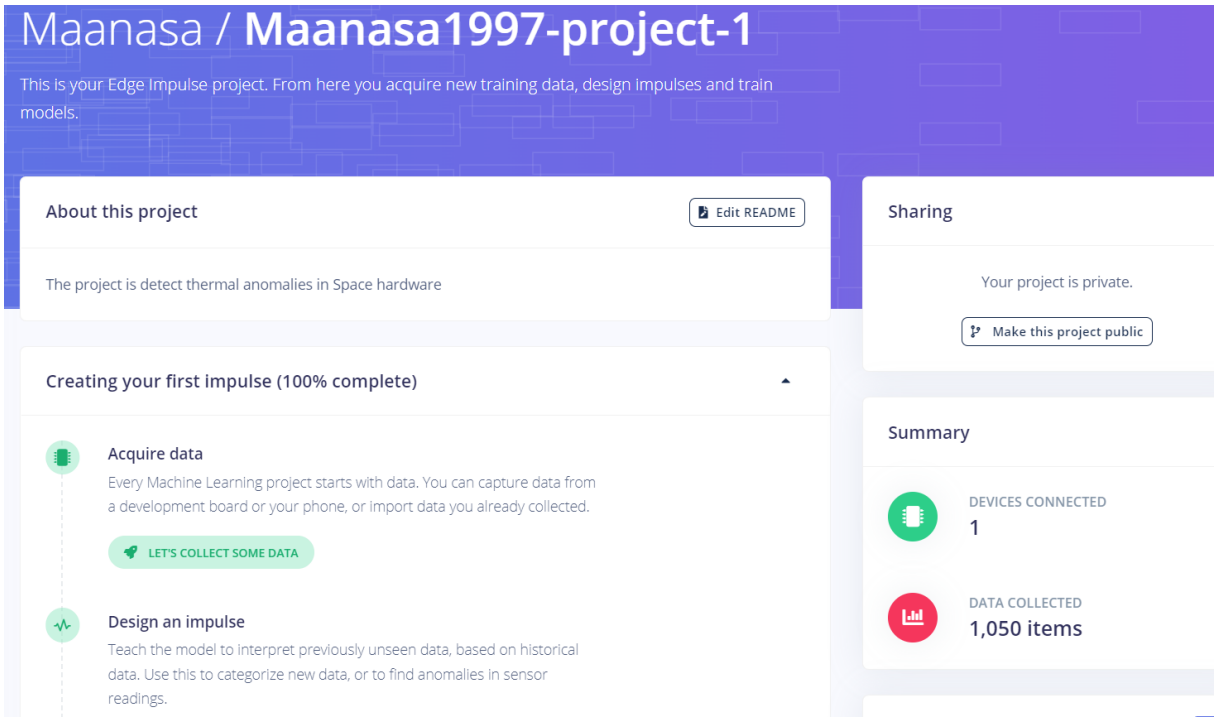

Figure 3.18: Creating a project on Edge Impulse

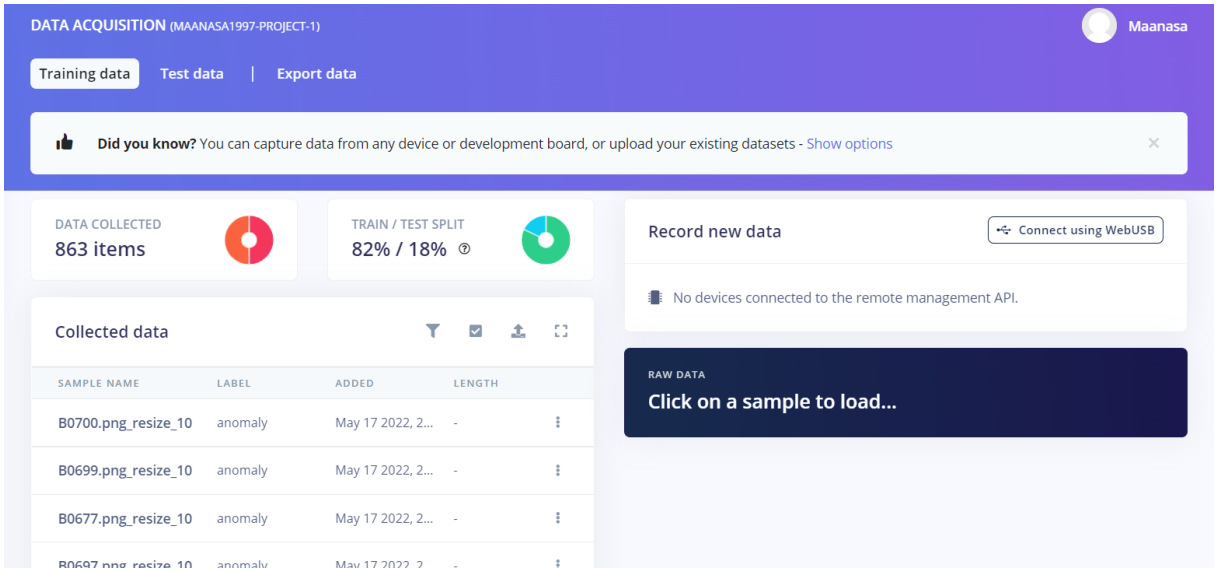

Figure 3.19: Training data collection

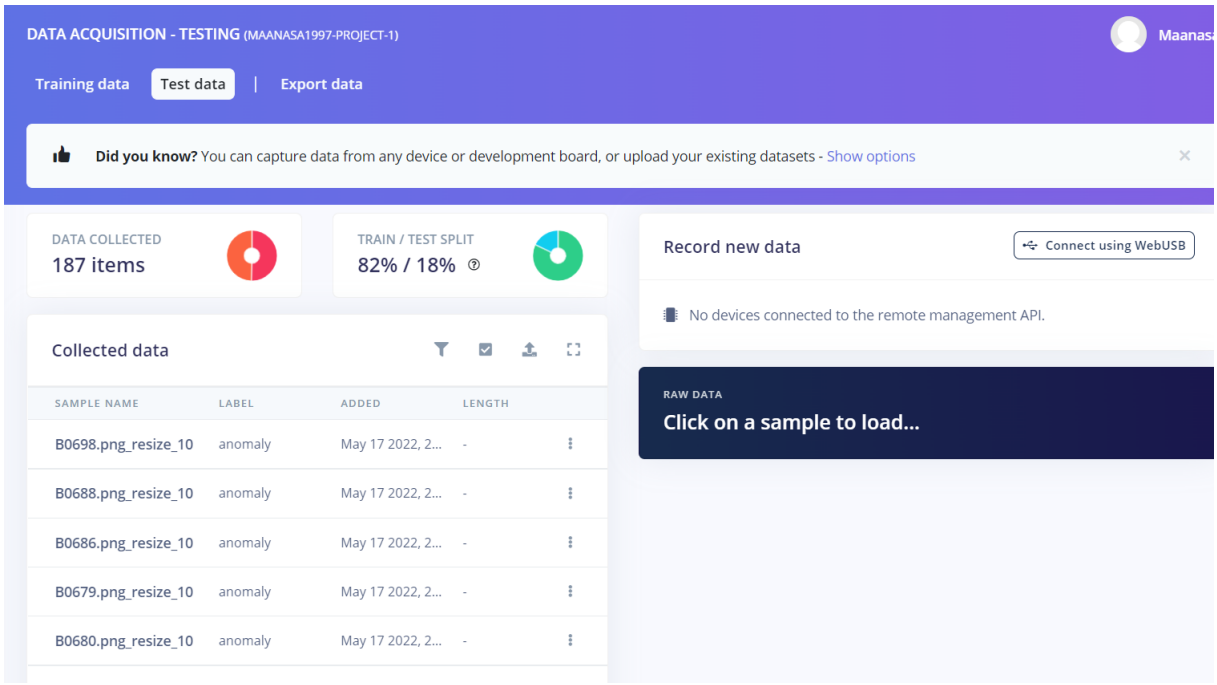

Figure 3.20: Test data collection

Data Preparation

This is the second step to building an AI model and in this step, the collected images are modified to increase the accuracy of the model. The Edge Impulse platform recommends resizing the input thermal images with a height of 96 and width of 96 and hence this step is performed.

The thesis uses transfer learning techniques for deep learning which involves using deep learning algorithms which are already trained using another data set. The use of pre-trained models can increase the performance of the new AI model, and reduce the time and number of labelled data required to train the new model. The choice of the transfer learning algorithm is explained in the next subsection.

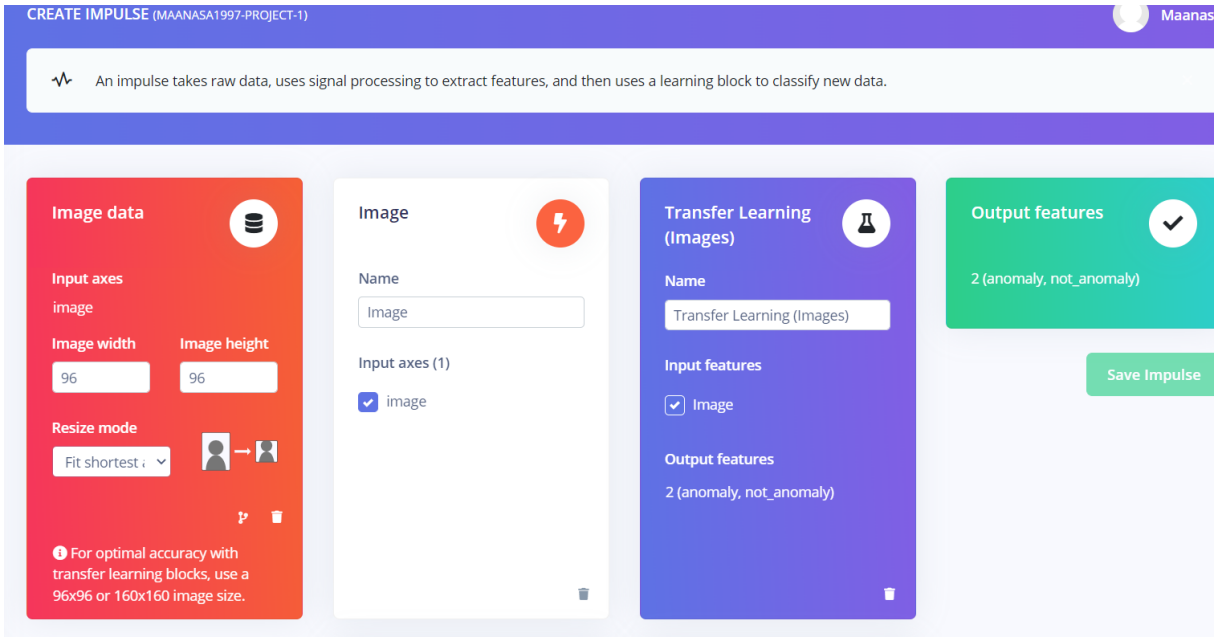

Figure 3.21: Data preparation using Edge Impulse

The data visualization block in the edge impulse platform is used to check if all the thermal images are labelled correctly as anomalous and normal images and if they are RGB images. The training and test data were labelled since supervised learning algorithms are used.

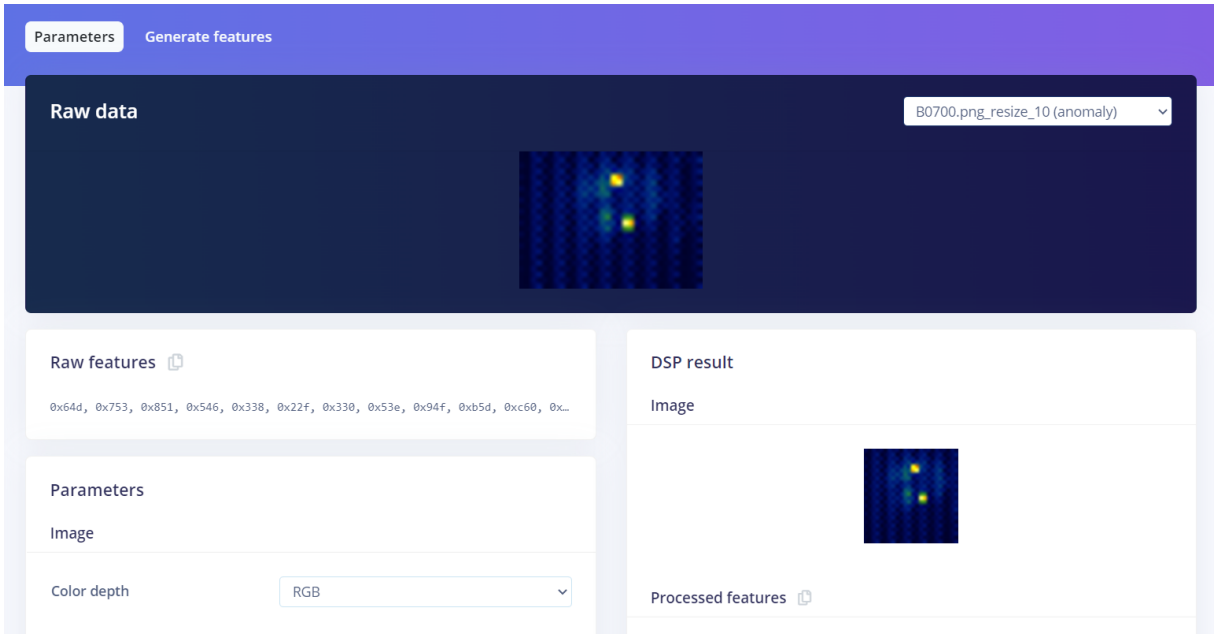

Figure 3.22: Data visualization

Build an AI model

In this step, the AI model is chosen and implemented. Edge Impulse uses the TensorFlow ecosystem for training, optimizing, and deploying deep learning models to embedded devices. Fig. 3.23 shows the python implementation of the MobileNetV2 architecture using TensorFlow's Keras APIs integrated on the Edge impulse platform. The code is available in appendix A.1. The pre-trained MobileNetV2 model is implemented; transfer learning is applied as the model was trained using ImageNet data. Edge Impulse also provides the possibility to visualise the latency, RAM and ROM usage for MobileNetV2 architecture for RGB images as shown in Fig. 3.24 and Fig. 3.25.

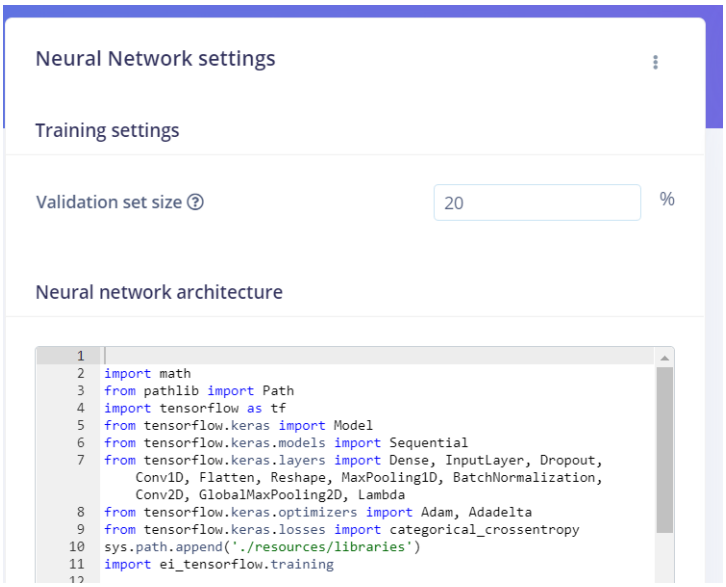

Figure 3.23: Tensorflow code on the Edge Impulse platform

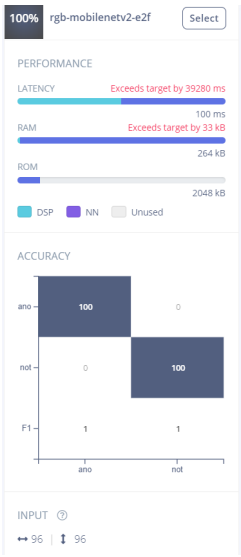

Figure 3.24: Finding optimal architecture using edge impulse: part 1

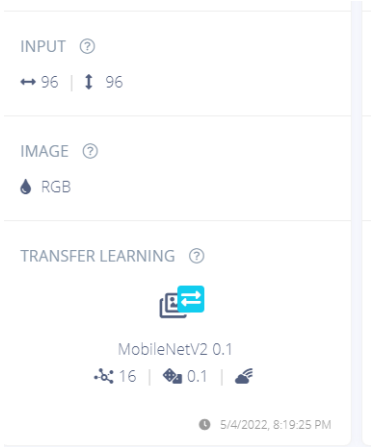

Figure 3.25: Finding optimal architecture using edge impulse: part 2

Train the AI model

In this step, the MobileNetV2 architecture is trained using the training data containing the thermal images. Fig. 3.26 shows that 100% accuracy is obtained by training the model with new data. Thus, the AI model is able to successfully differentiate between anomalous and normal images. The batch size is 32 and 20 epochs were used for training the data.

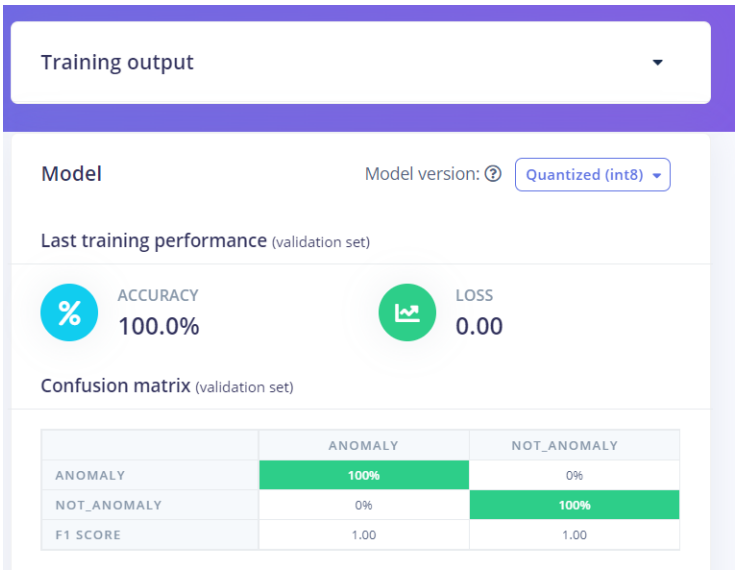

Figure 3.26: Model training performance

The Edge impulse compiler also provides a feature explorer to visualize the input thermal images in one 3D graph. The axes help visualize the training data classified by the neural network. This is illustrated in Fig. 3.27. A rough estimate of the time taken, peak ram usage and flash usage of the MobileNetv2 architecture on the Raspberry Pi are also provided to verify if the model is suitable for deployment.

Figure 3.27: Edge Impulse feature explorer

Test the AI model

After a good performance of the MobileNetV2 architecture on the training data, the architecture is validated by providing test data as input. Fig. 3.28 shows that 100% accuracy is achieved and the architecture is able to successfully classify the anomalous and normal images for test data. This step confirms that MobileNetV2 architecture is suitable for deploying on the embedded device.

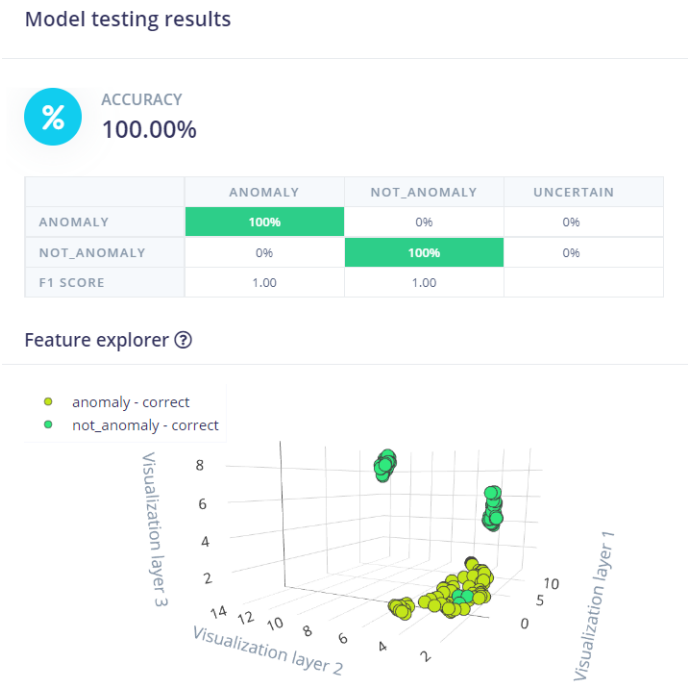

Figure 3.28: Model test performance

Deploy the AI model

After training and testing the model, the model has to be deployed on the Raspberry Pi Zero; the computer module for the compact infrared payload.

It is important to note that training the deep learning algorithm on the Raspberry Pi is difficult due to low computation resources such as memory (RAM, ROM), speed and total cores. The thesis uses AI inference which allows detection of thermal anomalies in the thermal image using trained AI models that are deployed on the embedded device. AI inference allows trained deep learning algorithms to classify new real-time data generated by the embedded device. The neural networks deployed using Edge Impulse are int8 quantized and this means the precision of the model weights is reduced from float32 values to reduce memory and computing requirements.

Edge Impulse platform allows building an executable file in .eim format which contains the AI code, IPC layer (over a Unix socket) and optimization for the processor. The thesis used this .eim format file to run AI models using real-time data as it is a self-contained file and it avoids installing a specific version of TensorFlow and python dependencies for the Raspberry pi Zero. This also reduced the development time of deploying the AI model

on the payload. The size of the downloaded model.eim file is 10MB. The .eim format file is then run using the Edge Impulse Linux runner on the Raspberry pi Zero. The results of the real-time performance of AI detection software on the Raspberry pi Zero are mentioned in the next section.

3.6 Result of AI techniques to detect thermal anomaly

The following section focuses on testing the AI detector or the thermal anomaly detection software which is implemented on the breadboard model of the payload shown in Fig. 3.29.

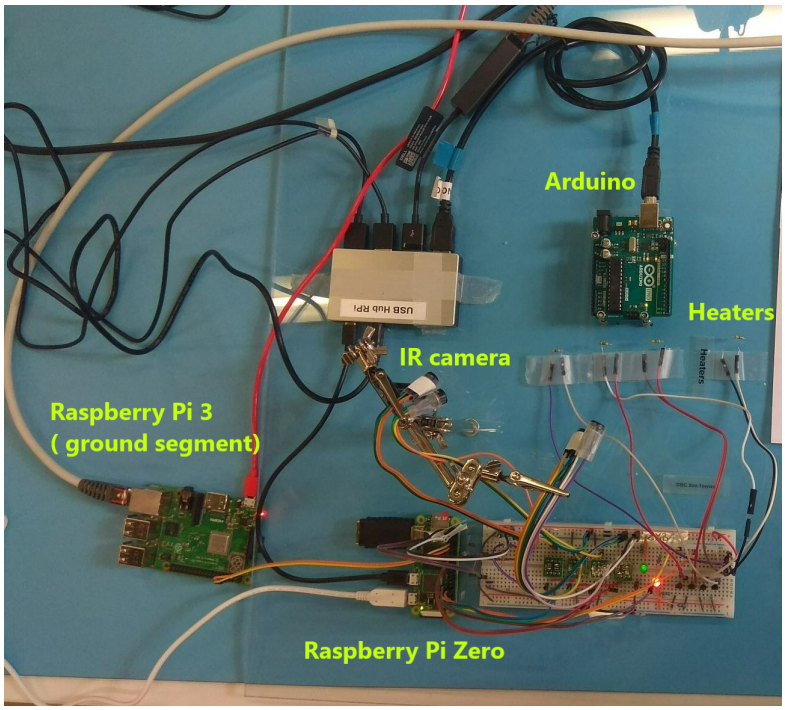

Figure 3.29: Breadboard model of payload

The payload is tested in real-time using the web interface and the hardware setup is the same as indicated in Fig. 3.13. The hardware setup should not be disturbed as the AI models were trained using the setup. The following observations are made on the output of the thermal anomaly detection software:

1. The AI detector software is able to successfully classify the anomalous and normal images. Fig. 3.30 displays the web interface with captured thermal images when no heater is turned on. The web interface shows that normal operation is detected with a probability of 1.0 (Fig. 3.32). Fig. 3.33 displays the web interface with captured thermal images when two heaters are turned on. The web interface shows that an anomaly is detected with a probability of 0.9 (Fig. 3.35).

2. There is some delay in classifying the thermal images due to the computation time of AI detector software which is approx 30 ms (milliseconds) and due to the serial communication delay between ground segment and payload. Hence, the software

must be run for more than a 30 ms duration to detect thermal anomalies on the captured thermal images.

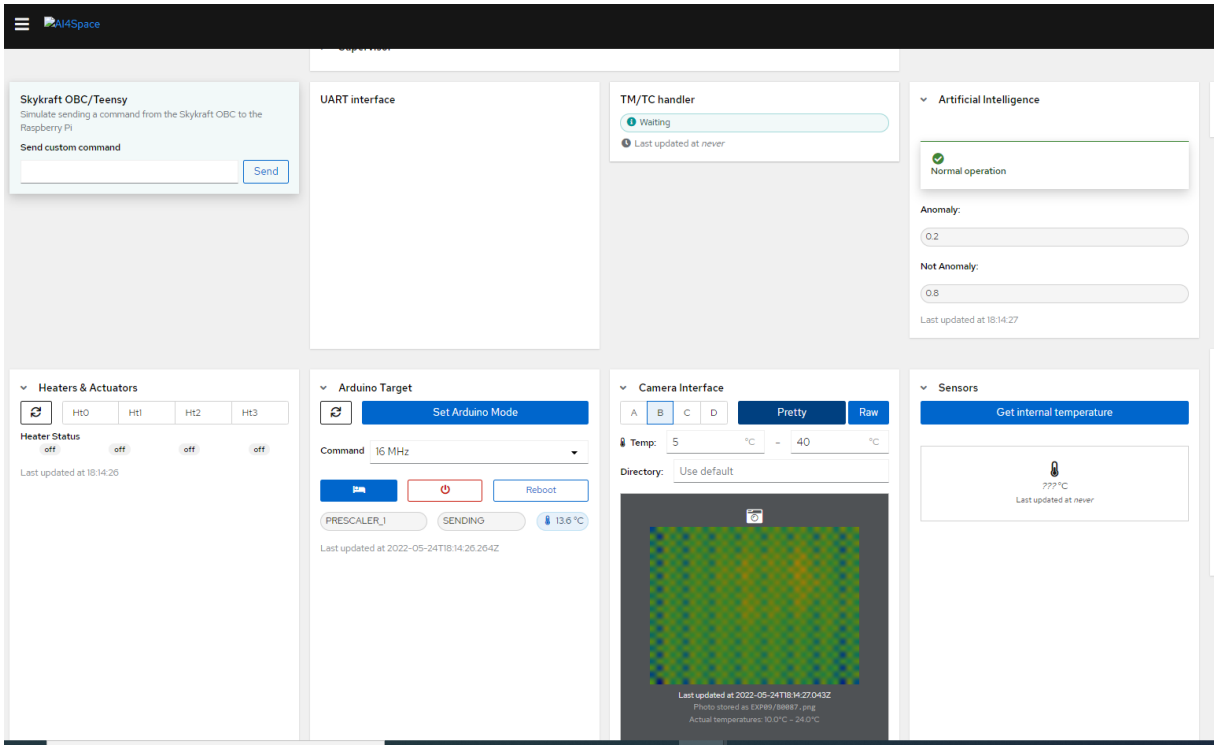

Figure 3.30: Web interface during normal operation

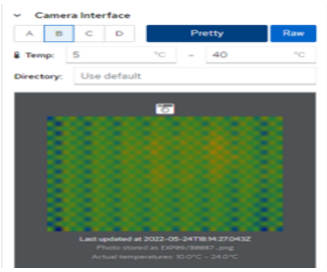

Figure 3.31: Scenario with normal operation

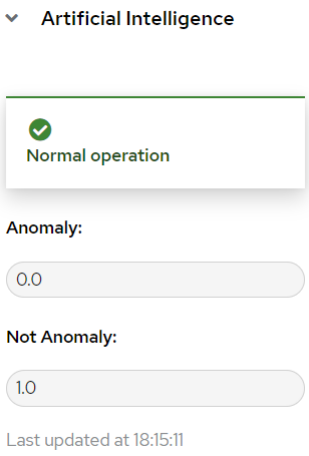

Figure 3.32: Normal operation detected

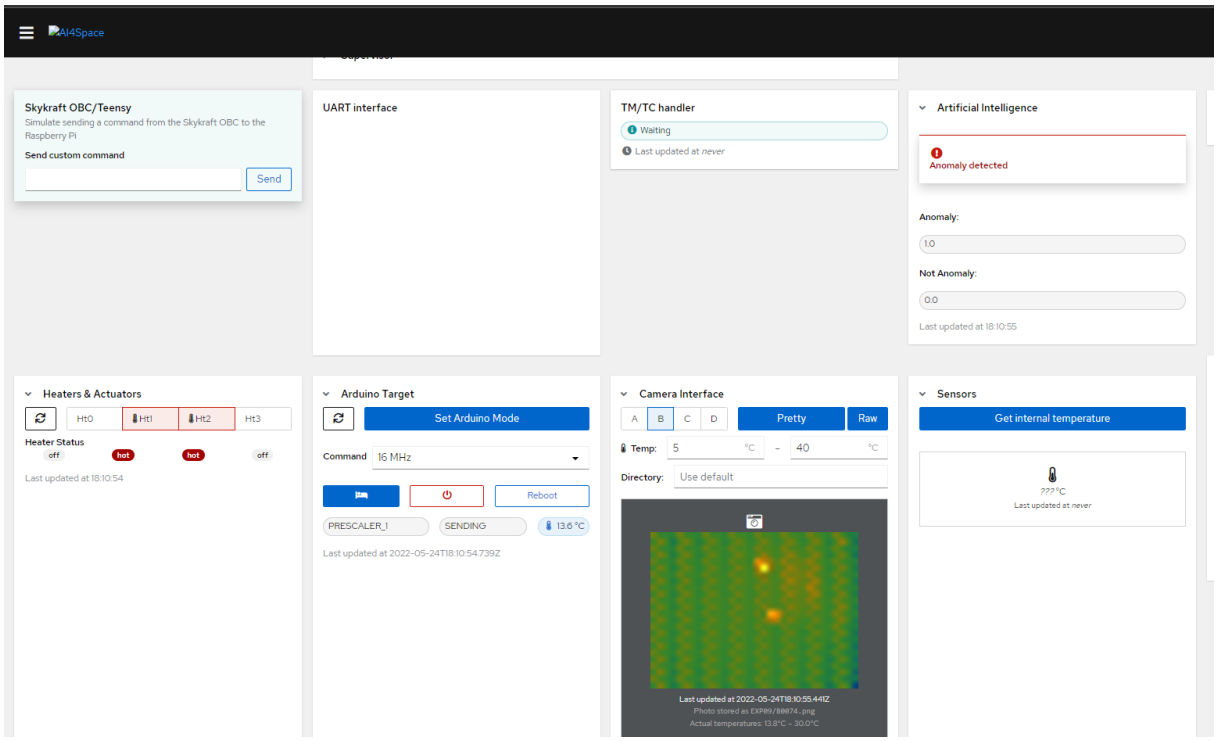

Figure 3.33: Web interface when anomaly is detected

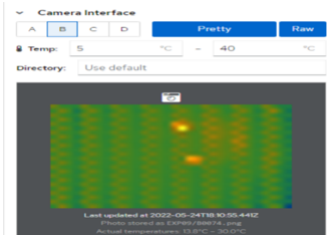

Figure 3.34: Scenario with anomaly detection

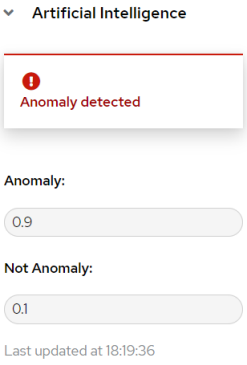

Figure 3.35: Anomaly detected

3.7 Testing the software architecture

The previous sections showed the embedded software implementation on the breadboard model of the payload (shown in Fig. 3.29). This section focuses on the unit testing of software which helps to test isolated individual software components and each component is tested for correctness. This testing was carried out by the software lead of the AI4Space mission. However, it is important to have external testers who can create unit test cases and provide an external point of view on the software under test. It can also help detect errors that a developer might have missed and ensure the increased quality of the software. Hence, this thesis contributed to testing the software architecture of the payload. An important point to note is that the embedded software unit tests are affected by the hardware in use and they might not be exhaustive. [37] [38]

The Robot framework [13] is an open-source automation framework that is used for test automation in the AI4Space mission. The unit test cases were used to verify the operation of the payload, for example, health, the internal temperature of the onboard computer, large packet handling etc. These unit test cases are created to ensure that the payload will respond to telecommands sent from the ground segment and to verify if the telemetry received is the expected response. The testing framework is run on the ground segment to simulate the interaction between the ground segment and payload. The serial interface which uses the UART communication protocol is used to send telecommands from the Raspberry Pi 3 in the ground segment to the Raspberry Pi Zero of the payload. However, there is also a possibility to test the embedded software components without the serial interface using RabbitMQ which is not discussed in this thesis.

This thesis contributed to the unit test cases for verifying the response received after sending the telecommands for file operations, for example, *file read*, *file list*, *file new* etc. Different files were created for each unit test operation to ensure the software components are isolated and do not depend on each other, one such example is, a new file is created before testing the *file read* telecommand and a different file is created for testing the *file append* telecommand. The different scenarios for the operation to fail were considered (for example, the file existence is verified before reading) and each telemetry received was verified.

These tests were created using the Robot Framework and an example output is shown in Fig. 3.36 and Fig. 3.37 where multiple unit test cases are run sequentially. It can be observed from the figures that some tests failed during the testing process and the telemetry received is verified using the test log shown in Fig. 3.38. Fig. 3.36 shows that a Timeout error occurred while running a unit test and there are helpful and detailed messages for debugging the issue. Therefore, testing using the Robot Framework helped catch these issues and they were successfully resolved to pass these test cases.

```
maanasa@pp1982:/mnt/c/Users/maanasa.sachidanand/test campaign/tests$ python3 -m robot --loglevel DEBUG --test "*" test.robot
[ WARN ] Not running on a Raspberry pi. Some modules may not work.
==============================================================================
Test :: Example test cases using the keyword-driven testing approach.
==============================================================================
Parameter: Temperature                                                | PASS |
------------------------------------------------------------------------------
Parameter: Boot Count                                                 | PASS |
------------------------------------------------------------------------------
Parameter: Health                                                     | PASS |
------------------------------------------------------------------------------
Ping without content                                                  | PASS |
------------------------------------------------------------------------------
Ping with content                                                     | PASS |
------------------------------------------------------------------------------
Ping file_new                                                         | PASS |
------------------------------------------------------------------------------
Ping file_append                                                      | FAIL |
TimeoutError
------------------------------------------------------------------------------
[ WARN ] Received response for message None which was cancelled
Ping file_read_all case_1                                             | PASS |
------------------------------------------------------------------------------
Ping file_read_all case_2                                             | PASS |
------------------------------------------------------------------------------
Ping file_read_part case_1                                            | PASS |
------------------------------------------------------------------------------
Ping file_read_part case_2                                            | PASS |
------------------------------------------------------------------------------
```

Figure 3.36: Testing the software architecture: part 1

[13] https://robotframework.org/

```
Ping file_read_part case_2                                                    | PASS |
------------------------------------------------------------------------------
Ping file_move                                                                | PASS |
------------------------------------------------------------------------------
Ping file_list_glob case_1                                                    | PASS |
------------------------------------------------------------------------------
Ping file_list_glob case_2                                                    | PASS |
------------------------------------------------------------------------------
File file_list_glob                                                           | FAIL |
TimeoutError
------------------------------------------------------------------------------
Test :: Example test cases using the keyword-driven testing approach.         | FAIL |
15 tests, 13 passed, 2 failed
==============================================================================
Output:  /mnt/c/Users/maanasa.sachidanand/test campaign/tests/output.xml
Log:     /mnt/c/Users/maanasa.sachidanand/test campaign/tests/log.html
Report:  /mnt/c/Users/maanasa.sachidanand/test campaign/tests/report.html
maanasa@pn1982:/mnt/c/Users/maanasa.sachidanand/test campaign/tests$
```

Figure 3.37: Testing the software architecture: part 2

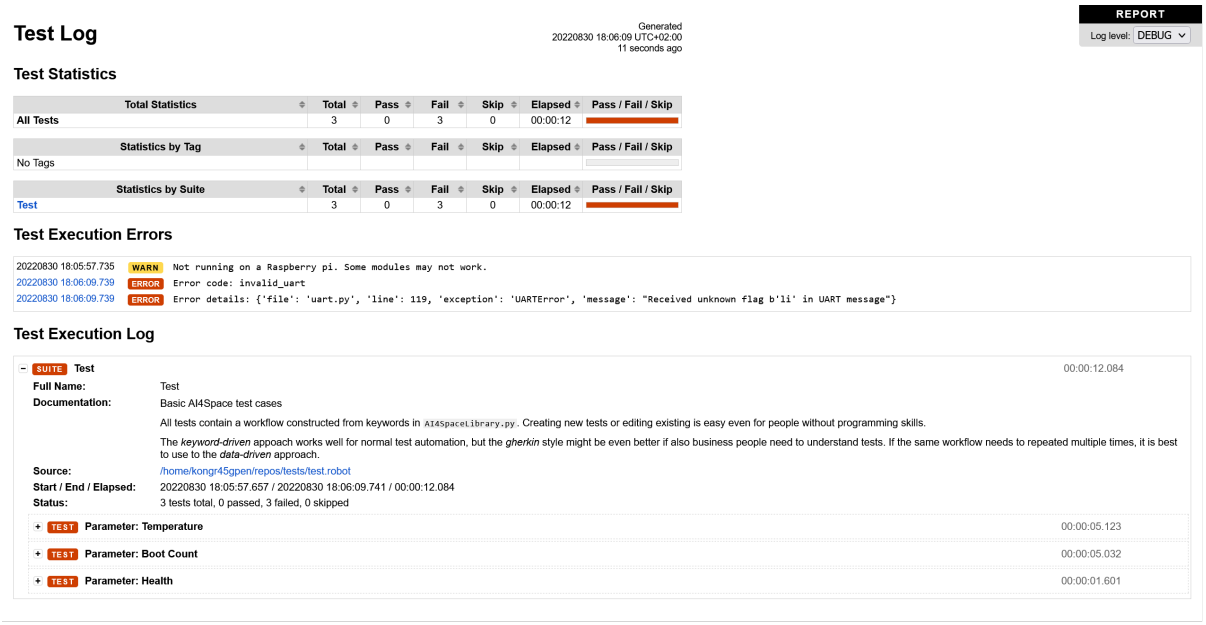

Figure 3.38: Test log

3.8 Assembly of the flight payload

The following section summarises the assembly of the flight payload after successful functional testing of the breadboard model of the payload. Before starting the assembly of the flight payload, the hardware components and tools required for assembly were ordered and PCB board design was completed.

The thesis contributed to documenting the assembly procedure. This involved the following steps:

- Recording all the hardware components, tools and consumables required for assembly of the flight model of payload shown in Fig. 3.40. For ex: USB microscope, Solder paste, 1k Resistor, 100n capacitor etc.

- Documenting the integration procedure step-by-step. More than 100 steps for integrating the hardware components on the PCB board of payload were documented. This required understanding the soldering challenges of components with different sizes and integrated circuit packages (ex: surface mount packages), verifying all solder points on the board and ensuring all 71 hardware components are included. The steps were also included for the IR camera board (shown in Fig. 3.41) which had to be mounted on top of the PCB board of flight model.
- Separate steps recorded for the flight model and the ground segment.

This work helped identify potential issues before causing permanent issues in the assembly process of payload such as identifying unordered components, components that did not have spares etc. The work also provided me with experience in the assembly process of space payloads.

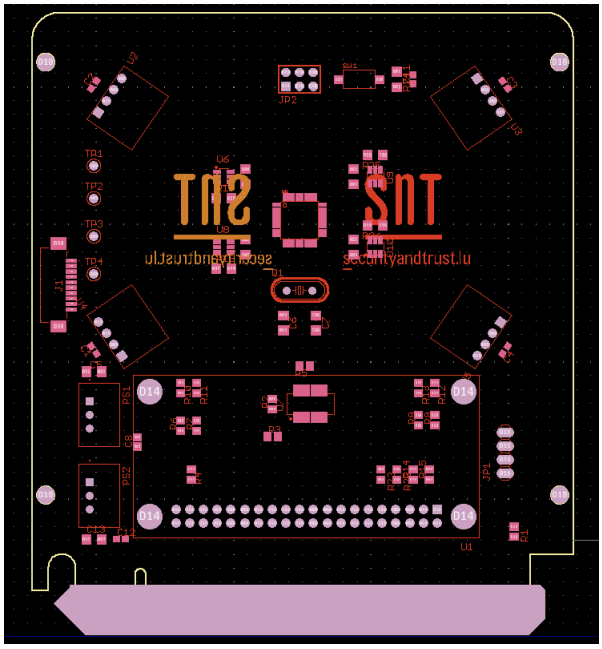

Figure 3.39: PCB layout of payload

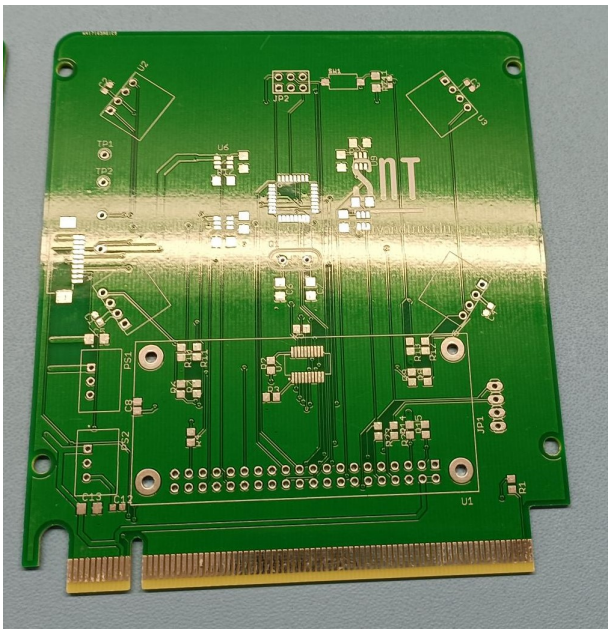

Figure 3.40: Flight model of payload before assembly

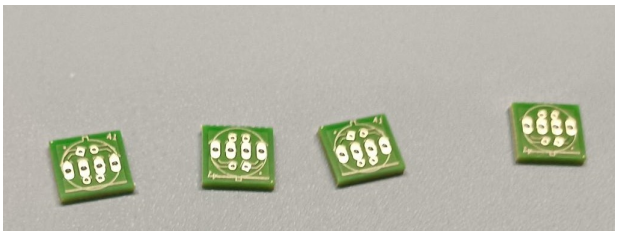

Figure 3.41: IR camera board

3.9 Challenges and Limitations

The AI4Space mission is a novel space mission and there were limited resources to verify the feasibility of the edge computing application to detect thermal anomalies during space flight. The project faced several challenges during project execution and some of the critical challenges are listed:

1. Delay in Assembly - The thesis could not test the AI algorithm on the flight model of the payload due to the delay in acquiring hardware components and chip shortage issues. Some components were replaced in the last phase of development due to the extended delay. It is important to consider this delay as an overhead for future space missions.

2. The space environment - It is difficult to simulate the harsh space environment on earth. It is also expensive in time and cost to do radiation tests. This leads to training the AI algorithms with data (thermal images) that is not accurate and it can lead to reduced performance of trained AI models in space.

3. Adapting to changes in space mission - The development of the payload started before the launch opportunity with SkyKraft. Hence, it was challenging to develop the payload adapting to the new mission launch date and spacecraft specifications such as interface, power and data rate. The changes in the downlink data rate and memory interface can reduce the performance of thermal anomaly detection since the AI models are highly reliant on the data that is fed to the model.

4. Ageing or Seasonal effects - The thesis did not consider the ageing effects of the electronics on the payload. Thus, It is important to train the models with recent telemetry to not detect ageing or seasonal effects as atypical behaviours. Since it leads to the chance of false alarm detection and correct detection of anomalies. One can reduce these effects by acquiring telemetry data during space flight.

5. Limitation of computing resources - The use of a Raspberry Pi leads to limitations on the choice of AI algorithm due to its limited memory and processor performance. It is only possible to do AI inference on these devices.

   To increase the performance of AI models in space flight, the following solutions can be implemented:

   (a) Increase the computation resources (ex: servers, MPSoC) in space

   (b) High downlink and uplink data rate - The AI models can be trained with the recent telemetry on the cloud in the ground segment and the newly trained model can be sent back to the payload.

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

The thesis accomplished the goals of developing an Edge AI application for compact infrared payload and also contributed to the development of space systems end-to-end using the Fast development approach.

4.1 Summary

The different steps performed in the thesis for the development of the compact infrared payload are illustrated in Fig. 4.1. The thesis started with choosing a deep learning architecture for thermal anomaly detection. Then, the system modelling of payload was done using MBSE methodology. This was followed by modifying the software to integrate the AI detector to detect the thermal anomalies on the target electronics of the payload. The MobileNetV2 architecture (i.e. supervised learning algorithm) was successfully trained, implemented and tested on the breadboard model of the payload. The results of the AI model showed 100% accuracy in detecting anomalous images from the captured thermal images for the breadboard model of payload. Then, the software architecture was tested for reliable communication with the ground segment by creating unit test cases. The final part of the thesis involved documenting the assembly procedure and this step helped detect potential issues and failures in the assembly process for the flight model of payload.

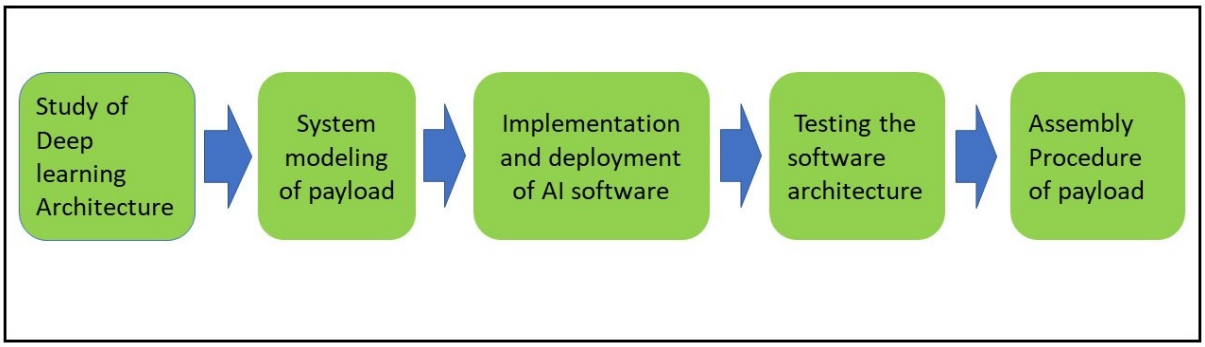

Figure 4.1: The different steps performed in this thesis

The thesis used the concepts of systems engineering, software development, electronics

and data science to develop edge computing in space applications. The different challenges and limitations faced during the fast development approach of the payload have been addressed and some solutions have been proposed.

4.2 Future Work

This thesis deployed and tested the AI algorithm on the breadboard model of payload. The future work will involve deployment and testing of the AI model on the flight model of the payload (shown in appendix. A.2). This step has to be conducted after the assembly process of the PCB board of flight model. It is also important to train the AI model with thermal images captured in the vacuum chamber and simulated space environments for increasing the accuracy of the model.

In case of low accuracy of a supervised learning algorithm (i.e. MobileNetV2 architecture), then the unsupervised learning algorithms using Autoencoders can be explored for detecting thermal anomalies on space electronics hardware.

4.2.1 MPSoC infrastructure for AI-driven payload

Due to the delay in assembly of the flight model of the payload. The preliminary work was continued for the second configuration of the payload with MPSoC as the onboard computer. The hardware used for implementing the AI models include: A baseboard namely Enclustra Mars EB1[1], the system on chip module namely Enclustra Mars XU3[2] and a heat sink. The MPSoC module is built around Xilinx Zynq Ultrascale+ MPSoC ARM quad-/dual-core Cortex-A53. The AI models can be implemented on this module using the Vitis AI infrastructure.

The following factors make it a better choice in comparison to the Raspberry Pi as an onboard computer:

- Lower power consumption
- Combination of a powerful CPU system with the parallel processing power and real-time capabilities of an FPGA system
- The availability of eMMC flash helps reduces mechanical vibrations during payload launch in comparison to using an SD card on Raspberry Pi.

[1] https://www.enclustra.com/en/products/base-boards/mars-eb1/
[2] https://www.enclustra.com/en/products/system-on-chip-modules/mars-xu3/

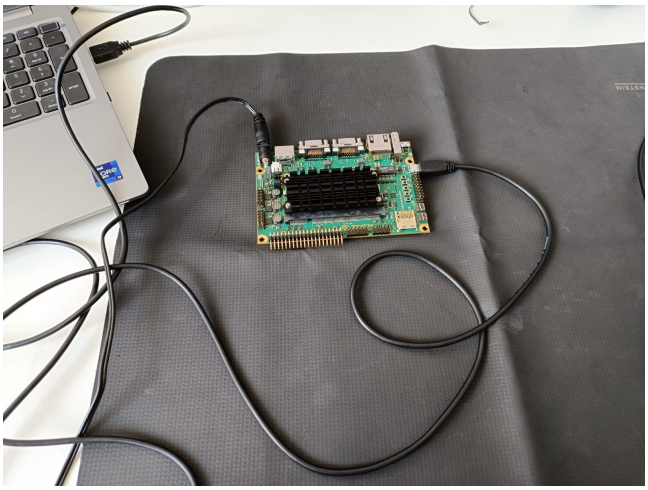

Figure 4.2: Enclustra Mars EB1 base board and Enclustra Mars XU3 board

4.2.2 Software tools and Installation

The following software tools are required for implementing the AI algorithm on the Xilinx Zynq Ultrascale+ MPSoC :

1. VirtualBox[3] - It is a virtualization software. It is used to create Linux virtual machine for running the Vitis AI and Petalinux tools. Check if the Ubuntu download version is compatible with the Vitis AI environment and also Enclustra boards (The payload uses Enclustra Mars XU3 system on chip module).
2. Docker[4] - It is used to access docker containers of Vitis-AI tools
3. Putty[5] - It is an open source software. It is used to connect to a serial port and communicate with the MPSoC.
4. Petalinux[6] - The PetaLinux tools help to customize, build and deploy Embedded Linux solutions on Xilinx processing systems.
5. Vitis AI[7] - The Vitis AI development environment is a specialized development environment for accelerating AI inference on Xilinx embedded platforms.

The Xilinx MPSoC devices have the advantage of using the Deep Learning Processor Unit (DPU [8]) which is a programmable engine dedicated for CNN. The MobileNetV2 architecture has to be deployed on this configuration of payload using the Vitis AI TensorFlow design process.

[3]https://www.virtualbox.org/
[4]https://www.docker.com/
[5]https://www.putty.org/
[6]https://www.xilinx.com/content/xilinx/en/products/design-tools/embedded-software/petalinux-sdk.html
[7]https://github.com/Xilinx/Vitis-AI
[8]https://www.xilinx.com/products/intellectual-property/dpu.html

REFERENCES

[1] Les actualités. La société de canberra skykraft se prépare à lancer des satellites à bord de la fusée spacex. lesactualites.news/affaires/la-societe-de-canberra-skykraft-se-prepare-a-lancer-des-satellites-a-bord-de-la-fusee-spacex/, January 2022.

[2] Google AI. Mobilenetv2: The next generation of on-device computer vision networks. ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html, April 2018.

[3] Google AI. Efficientdet: Towards scalable and efficient object detection. https://ai.googleblog.com/2020/04/efficientdet-towards-scalable-and.html, April 2020.

[4] Philippe Perdu. Failure analysis on space electronics: Best practices, challenges and trends. In *2018 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, pages 1–6, 2018.

[5] Daniel M. Fleetwood, Peter S. Winokur, and Paul E. Dodd. An overview of radiation effects on electronics in the space telecommunications environment. *Microelectronics Reliability*, 40(1):17–26, 2000.

[6] Sylvain Fuertes, Gilles Picart, Jean-Yves Tourneret, Lotfi Chaari, André Ferrari, and Cédric Richard. Improving spacecraft health monitoring with automatic anomaly detection techniques. In *14th international conference on space operations*, page 2430, 2016.

[7] ESA. New ϕ-lab in sweden and the rise of edge computing in space. commercialisation.esa.int/2022/05/new-%CF%86-lab-in-sweden-and-the-rise-of-edge-computing-in-space/, May 2022.

[8] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S. Nikolopoulos. Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 20–26, 2016.

[9] Gianluca Giuffrida, Luca Fanucci, Gabriele Meoni, Matej Batix010D;, Lx00E9;onie Buckley, Aubrey Dunne, Chris van Dijk, Marco Esposito, John Hefele, Nathan Vercruyssen, Gianluca Furano, Massimiliano Pastena, and Josef Aschbacher. The amp;x03a6;-sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

[10] IBM. The new space age: Ibm develops a unique, custom edge computing solution in space. www.ibm.com/cloud/blog/ibm-develops-a-unique-custom-edge-computing-solution-in-space, February 2021.

[11] Vivek Kothari, Edgar Liberis, and Nicholas D. Lane. The final frontier: Deep learning in space. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, HotMobile '20, page 45–49, New York, NY, USA, 2020. Association for Computing Machinery.

[12] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Lx00E9;onie Buckley, Mihalis Psarakis, Kay-Obbe Voss, and Luca Fanucci. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, 35(12):44–56, 2020.

[13] Nvidia. What is edge ai and how does it work? by tiffany yeung. https://blogs.nvidia.com, February 2022.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] Yuanyuan Sun, Lili Guo, Yongming Wang, Zhongsong Ma, and Yi Niu. Fault diagnosis for space utilisation. *The Journal of Engineering*, 2019, 12 2019.

[16] Maciej Ziaja, Piotr Bosowski, Michal Myller, Grzegorz Gajoch, Michal Gumiela, Jennifer Protich, Katherine Borda, Dhivya Jayaraman, Renata Dividino, and Jakub Nalepa. Benchmarking deep learning for on-board space applications. *Remote Sensing*, 13(19):3981, 2021.

[17] Musab Coşkun, Özal YILDIRIM, UÇAR Ayşegül, and Yakup Demir. An overview of popular deep learning methods. *European Journal of Technique (EJT)*, 7(2):165–176, 2017.

[18] Shaveta Nagpal, Munish Kumar, · Maruthi, Maruthi Rohit Ayyagari, and · Kumar. A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 07 2019.

[19] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[20] Cai Lile and Li Yiqun. Anomaly detection in thermal images using deep neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2299–2303, 2017.

[21] Gwanyong Park, Minhyung Lee, Hyangin Jang, and Changmin Kim. Thermal anomaly detection in walls via cnn-based segmentation. *Automation in Construction*, 125:103627, 2021.

[22] Daniel Pérez-Aguilar, Redy Risco-Ramos, and Luis Casaverde-Pacherrez. Transfer learning for binary classification of thermal images. *Ingenius. Revista de Ciencia y Tecnología*, (26):71–86, 2021.

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[24] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[25] Giovanni Cipriani, Donatella Manno, Vincenzo Di Dio, and Marzia Traverso. Thermal anomalies detection in a photovoltaic plant using artificial intelligence: Italy case studies. In *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*, pages 1–6, 2021.

[26] Giovanni Cipriani, Donatella Manno, Vincenzo Di Dio, and Giovanni Sciortino. Automatic detection of thermal anomalies in induction motors. In *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*, pages 1–6, 2021.

[27] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. 2019.

[28] Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, and Ivan Oleksiyuk. Autoencoders for unsupervised anomaly detection in high energy physics. *Journal of High Energy Physics*, 2021(6):1–32, 2021.

[29] Christoph Baur, Stefan Denner, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Autoencoders for unsupervised anomaly segmentation in brain mr images: A comparative study, 2020.

[30] Vincent L PISACANE. *Fundamentals of space systems*. Johns Hopkins University Appli, 2005.

[31] Miguel A Aguirre. *Introduction to space systems: design and synthesis*, volume 27. Springer Science & Business Media, 2012.

[32] Bruce Powel Douglass. *Agile systems engineering*. Morgan Kaufmann, 2015.

[33] K. J. Forsberg R. D. Hamelin D. D. Walden, G. J. Roedler and T. M. Shortell. Incose systems engineering handbook: A guide for system life cycle processes and activities, 4th edition. Wiley Online Library, 2015.

[34] Raspberry Pi. Raspberry pi zero 2 w. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/.

[35] Melexis. Far infrared thermal sensor array (32x24 res). https://www.melexis.com/en/product/MLX90640/Far-Infrared-Thermal-Sensor-Array.

[36] Afshin Gholamy, Vladik Kreinovich, and Olga Kosheleva. Why 70/30 or 80/20 relation between training and testing sets: a pedagogical explanation. 2018.

[37] Luciano Baresi and Mauro Pezzè. An introduction to software testing. *Electronic Notes in Theoretical Computer Science*, 148(1):89–111, 2006. Proceedings of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques (FoVMT 2004).

[38] Hua-ming Qian and Chun Zheng. A embedded software testing process model. In *2009 International Conference on Computational Intelligence and Software Engineering*, pages 1–5, 2009.

APPENDIX

A.1 TensorFlow code for MobileNetV2 architecture

```
import math
from pathlib import Path
import tensorflow as tf
from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv1D,
    Flatten, Reshape, MaxPooling1D, BatchNormalization, Conv2D,
    GlobalMaxPooling2D, Lambda
from tensorflow.keras.optimizers import Adam, Adadelta
from tensorflow.keras.losses import categorical_crossentropy
sys.path.append('./resources/libraries')
import ei_tensorflow.training

WEIGHTS_PATH = './transfer-learning-weights/edgeimpulse/MobileNetV2.0_1
    .96x96.color.bsize_64.lr_0_05.epoch_498.val_loss_3.85.hdf5'

INPUT_SHAPE = (96, 96, 3)


base_model = tf.keras.applications.MobileNetV2(
    input_shape = INPUT_SHAPE, alpha=0.1,
    weights = WEIGHTS_PATH
)

base_model.trainable = False

model = Sequential()
model.add(InputLayer(input_shape=INPUT_SHAPE, name='x_input'))
# Don't include the base model's top layers
last_layer_index = -3
model.add(Model(inputs=base_model.inputs, outputs=base_model.layers[
    last_layer_index].output))
model.add(Reshape((-1, model.layers[-1].output.shape[3])))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(classes, activation='softmax'))

```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),
                loss='categorical_crossentropy',
                metrics=['accuracy'])



# Implements the data augmentation policy
def augment_image(image, label):
    # Flips the image randomly
    image = tf.image.random_flip_left_right(image)

    # Increase the image size, then randomly crop it down to
    # the original dimensions
    resize_factor = random.uniform(1, 1.2)
    new_height = math.floor(resize_factor * INPUT_SHAPE[0])
    new_width = math.floor(resize_factor * INPUT_SHAPE[1])
    image = tf.image.resize_with_crop_or_pad(image, new_height,
   new_width)
    image = tf.image.random_crop(image, size=INPUT_SHAPE)

    # Vary the brightness of the image
    image = tf.image.random_brightness(image, max_delta=0.2)

    return image, label

train_dataset = train_dataset.map(augment_image, num_parallel_calls=tf.
   data.AUTOTUNE)

BATCH_SIZE = 32
EPOCHS = 20
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder
   =False)
callbacks.append(BatchLoggerCallback(BATCH_SIZE, train_sample_count,
   epochs=EPOCHS))
model.fit(train_dataset, validation_data=validation_dataset, epochs=
   EPOCHS, verbose=2, callbacks=callbacks)

print('')
print('Initial training done.', flush=True)

# Epochs to tune the model
FINE_TUNE_EPOCHS = 10
# percentage of base model to tune
FINE_TUNE_PERCENTAGE = 65

print('Fine-tuning best model for {} epochs...'.format(FINE_TUNE_EPOCHS)
   , flush=True)
# Load best model from initial training
model = ei_tensorflow.training.load_best_model(BEST_MODEL_PATH)

# Determine which layer to begin fine tuning at
model_layer_count = len(model.layers)
fine_tune_from = math.ceil(model_layer_count * ((100 -
   FINE_TUNE_PERCENTAGE) / 100))

# the entire base model is trained
model.trainable = True
```

```
# Freeze all the layers before the 'fine_tune_from' layer
for layer in model.layers[:fine_tune_from]:
    layer.trainable = False

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.000045),
                loss='categorical_crossentropy',
                metrics=['accuracy'])

model.fit(train_dataset,
                epochs=FINE_TUNE_EPOCHS,
                verbose=2,
                validation_data=validation_dataset,
                callbacks=callbacks,
                class_weight=None
            )
```

Listing 1: The TensorFlow code using the Edge Impulse platform

A.2 The progress in assembly of flight model of payload

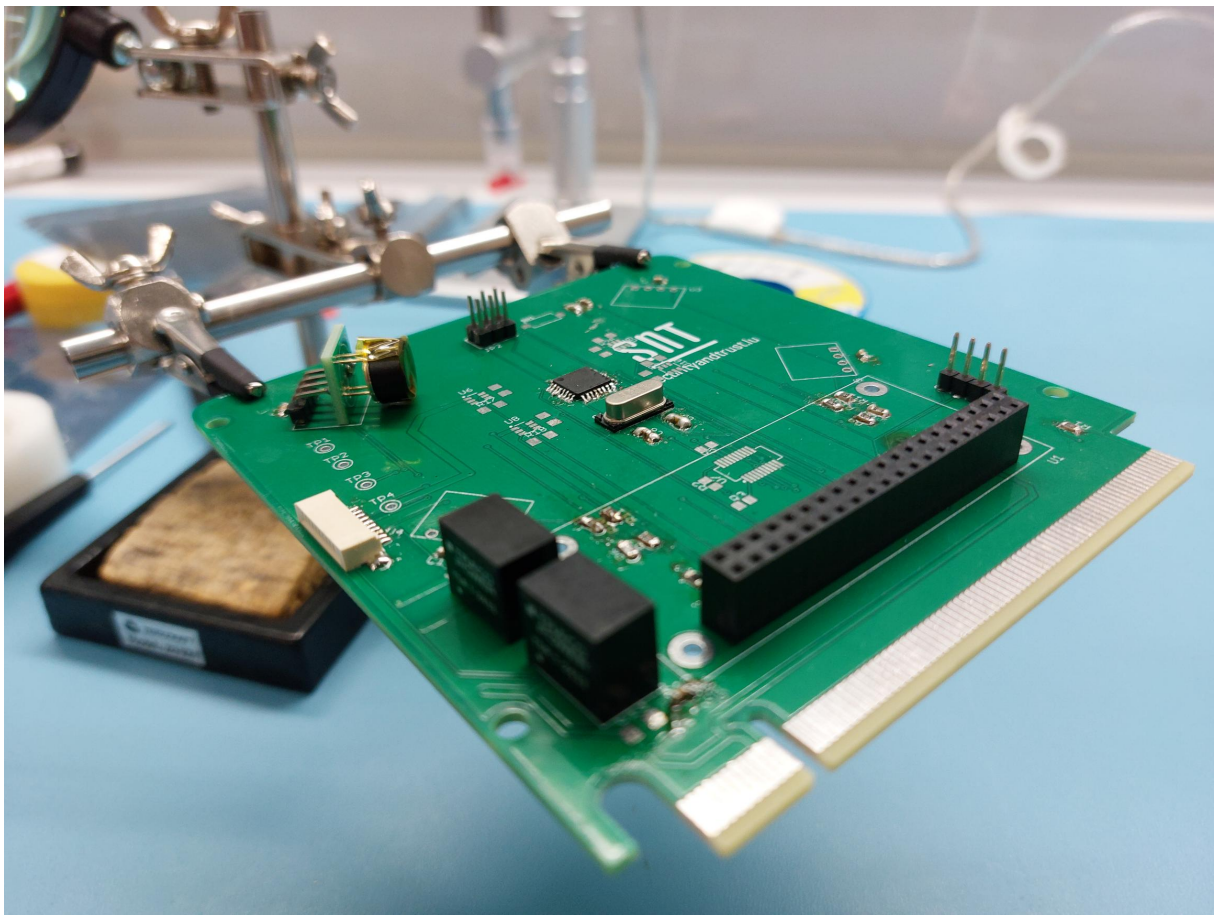

Figure 3: The assembly of flight model of the payload