UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Computer Science Curriculum

Hashim Hashimov

# Modelling Energy Consumption using Thermal Imaging

Master Thesis (30 EAP)

*Supervisor: Huber Flores*

TARTU, 2022

# Abstract

We contribute a novel sensing solution that can estimate the energy consumption of applications running in personal devices, such as smart and wearable devices, without requiring the instrumentation of the device itself. Energy estimation of applications running in these devices has become a complex challenge. Indeed, personal devices no longer have a detachable battery, such that their design can be further optimized to adjust better to the everyday activities of users. This in turn, it makes difficult to profile the energy consumption of software developed for these devices with typical solutions, like the Monsoon power meter. Our solution uses thermal imaging to derive energy consumption measurements while the applications are running in the device. We develop a functional prototype to demonstrate the feasibility of our solution. Through rigorous benchmarks that take into consideration different applications, we found that our solution can produce estimations of energy consumption that are comparable to existing hardware solutions. We also found that these estimations also capture differences between different types of applications. In addition, we also found that it is possible to use our solution to estimate the energy consumption of applications running in IoT devices. We also developed a regression model that predicts the estimated power from thermal imaging.

**CERCS:** P170 Computer science, thermal, power, energy

**Keywords:** thermal imaging, image analysis, computing energy, pervasive computing, energy prediction solutions, IoT

**Inimesest eraldunud soojusliku kiirguse kasutamine**

**kokkuvõte:** Meie pakume uut lahendust rakenduste energiakasutuse mõõtmiseks isiklikes seadmetes, ilma seadmeid muutmata. See hõlmab nii nuti kui ka kantavaid seadmeid. Tänapäeval on raske täpselt hinnata, kui palju energiat rakendused kasutavad. Kuna seadmetel puudub eemaldatav aku, siis see võimaldab nende igapäevast tööd optimeerida, aga tagajärjeks on nüüd raskem arvutada rakenduste energiakasutust. Üks põhjustest on lahenduste kasutamine tüüpiliste seadmete jaoks, näiteks Monsoon'i energia arvutaja, mitte seadme spetsiifilist lahendust. Meie toode kasutab termograafiat, millega me mõõdame rakenduste energiakasutust seadmes. Me arendame funktsionaalset prototüüpi, mis demonstreerib meie lahenduse lihtsust ja mugavust. Võrreldes teiste erinevate tarkvaradega leidsime, et meie lahendus annab sarnast energiakasutuse hinnangud nagu olemasolevad riistvara lahendused. Veel leidsime, et meie toode eristab rakendusi omavahel ja teeb erinevaid hinnanguid selle järgi. Lisaks sellele meie lahendus saab hinnata energiakasutust ka asjade interneti (IoT) seadmetes. Me veel arendasime regressiooni mudeli, mis ennustab oodatavat võimsust termograafiast.

**CERCS:** P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

**Märksõnad:** termopildistamine, jäätmekäitlus, mobiilne andmetöötlus, kõikehõlmav andmetöötlus, ringlussevõtu lahendused, IoT

# Contents

# List of Figures

## LIST OF FIGURES

# List of Tables

# Acknowledgements

Firstly, I would like to thanks to my family and friends who always motivated me and supported during this research. Also Prof. Dr. Huber Flores gave me detailed guidance, directions, suggestions which is really important help. Next, it is necessary to give my special thanks to Farooq Dar who always helped me during experiments, research and understanding the topic.

# 1

# Introduction

Estimation of energy consumption from smart, IoT, and wearable devices is of critical importance for the development of new applications. Estimating energy consumption is required for different applications, such as code smell analysis (2), code optimization, and refactoring (3). This is to avoid applications from becoming power hungry in devices with constrained resources. Unfortunately, estimation of energy has become a complex matter for the latest devices (4). Indeed, the battery from these devices is no longer detachable due to better usable designs and miniaturization (5). This poses a big challenge for developers as common solutions like the Monsoon power meter can no longer be used.

Besides specialized hardware solutions, other alternatives also require the instrumentation of the device (by removing the battery), e.g., Multi-meter, such that it is possible to measure energy consumption (6). Software solutions also have been proposed to overcome the problem. For instance, software profiles can either perform static code analysis on applications (7) or collect performance metrics of applications during run-time to estimate energy consumption (8). While these solutions can provide a good enough estimation of energy consumption without instrumenting the devices, these solutions are prone to errors caused by the run-time execution of code. Indeed, code execution can change drastically depending on different factors, e.g., input parameters (9), concurrent applications (10) and intermittent network connectivity (11). Thus, new innovative solutions that can be used for estimation of energy are required.

In this thesis, we investigate the use of thermal imaging to estimate energy consumption from applications running in devices. Thermal imaging is typically utilized to

measure temperature in industrial applications by looking at light changes in the invisible spectrum (12). By linking temperature changes with energy consumption measurements from devices, we derive a regression model that can be used to approximate energy consumption from thermal imaging. Through a rigorous evaluation with controlled experiments that consider different applications, we found that thermal imaging can be used to estimate energy from applications with R-Squared ($R^2$) of 0.86 and $RMSE$ around 0.10. Our work paves the way toward a simple energy estimation method that can quantify energy just by taking a picture.

## 1.1 Outline

This thesis is structured as follows:

- Chapter 2 reviews the state-of-the-art and explains the core background concepts of the work, including thermal imaging, energy profiling, energy measurements as well as reviews current applications and researches that benefit from thermal imaging.
- Chapter 3 presents the feasibility analysis of the solution by illustrating that thermal imaging has the potential to estimate and model the energy consumption of smartphones.
- Chapter 4 describes our methodology in detail and presents the pipeline of the proposed solution. We also include the mathematical background behind the estimation of energy from thermal imaging.
- Chapter 5 describes the different testbeds and procedures that we employ in order perform rigorous experimentation. This chapter also includes the experimental testbed and procedure for baseline experiments used to validate our proposed solution.
- In Chapter 6, we present the results of our experiments. We also present the results of extracting input parameters for modelling and estimation of energy. The results of the model efficiency have also been presented.
- Chapter 7 discusses the potential application scenarios as well as limitations and future work scope.

- Chapter 8 presents the summary and conclusion of our work.

# 2

# State of the Art

We begin this Chapter by discussing the current state-of-the-art methods for energy measurements and existing profiling techniques available. We highlight their limitations and differences against our proposed method. After that, we then discuss thermal imaging technique in general as this is the core technique of this thesis. We shed some light on the general applications of thermal imaging. Lastly, we also discuss the applications of thermal imaging for energy modelling.

## 2.1 Energy Measurement and Power Profiling

There are various existing tools in the market and they are different from each other by the levels of information they provide. The current state-of-the-art energy measurement and power profiling techniques are tedious and involve a lot of setup procedures to actually measure the energy and power. These energy measurement and power profiling techniques can be broadly divided into two categories: (13).

- Hardware-Based Energy Measurement
- Software-Based Energy Measurement

### 2.1.1 Hardware-Based Techniques

Hardware-based measurements are extremely precise, nevertheless, a special setup is required to connect hardware components. This requires to bypass the battery of the device with the appliance, e.g., multi-meter, such that the power is provided by the

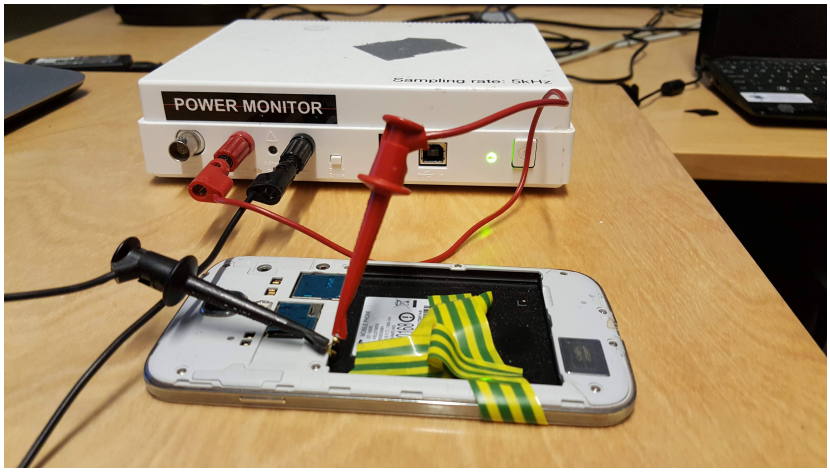



**Figure 2.1:** Hardware Based Energy Measurement using Monsoon Power Monitor

appliance rather than the battery (14). Monsoon Power meter (15) can be a good example of hardware-based tools (Figure 2.1. The common logic behind external tools is connecting the device and providing a voltage to a resistor which is connected to the power supply of the mobile phone. In this way, we can calculate the power by using changes in voltage. These hardware-based tools are accounted for as ground baseline values for software-based tools because of the accuracy.

Other hardware solutions include, Low-Energy Aware Platform (LEAP), which is a solution that can be used to measure consumed power in mobile apps (16). Besides being an easy and powerful tool, it can calculate the component-level energy consumption. Similarly, authors of (17) created a tool called PowerScope which is able to monitor energy usage. The main idea behind the tool is that PowerScope uses a digital multimeter and another machine that has control over a multimeter. That's why we have accounted PowerScope as a hardware-based approach since it requires hardware setup. Also, PowerScope monitors energy based on each process. All in all, PowerScope uses 3 components to sample energy consumption. Components are System Monitor, Energy Monitor, and Energy Analyzer. Even though it is a hybrid method, we have put it under hardware-based techniques.

Additionally, Intel implemented an SDK called Energy checker (18) which allows exporting counter from the application so that it can calculate time spent on each process and estimate power consumption. But, because of the hardware PowerMeter requirement, the tool is accounted as hardware-based. Likewise, the Greenminer framework

also can be used to measure energy consumption and to create automated reports for application developers (19). The framework containers functionalities to process the results by aggregating, analyzing, and visualizing.

Another recently studied (20) (21) popular method is Watts Up? Pro (22) which is a less expensive device to measure energy consumption. Watts Up? Pro is mainly designed to be connected between the power supply and the device that is being measured. One advantage is that Watts Up? Pro can perform monitoring while the main software is running. The Arduino Uno (23) electronic board also can be used to measure energy usage such that the GreenMiner testbed (19) enabled the use of Arduino by connecting to the device.

### 2.1.2 Software-Based Techniques

Software solutions are the alternative to hardware ones. Despite software-based measurement tools are not highly accurate (when compared with hardware-based ones), they can be easily installed in the smartphone and integrated into the development life cycle of applications to estimate power (24). It is important to mention that software-based tools are created by building mathematical models based on the hardware-based tools in order to map the consumption of each component into the software (25). Another way is watching the power changes during the execution of the application and find which one consumes more.

Several software-based solutions have been proposed over years. PowerTutor is a software-based tool that displays real-time consumption. PowerTutor developers built a model by creating various cases and comparing them to consumption results from the monsoon power meter. Similarly, PCE (Power Consumption Estimator) is an Android Studio plugin that predicts application consumption (26). It inspects the source code, as opposed to PowerTutor, to generate consumption estimates. It is also necessary to determine the application's use case in order to see the results.

Additionally, there are tools called PowerAPI (27), (28), and Jalen (29) which use power models for predicting energy usage for each process even Jalen can provide hardware-level energy metrics. PowerAPI can measure energy in real-time while Jalen is supposed to provide monitoring for energy consumption of components of software code such as methods, and classes. The authors calculated PowerAPI's accuracy as approximately $0.5 - 3\%$. Another popular software for energy measurement is Trepn (30)

which is officially created by Qualcomm and can provide profound information about the energy consumption of the devices which uses Qualcomm processors. Trepn is able to present data in real-time by gathering information from CPU statistics, power statistics, GPS, memory, battery, and more sensors. The tool enables testing power consumption for application developers however, it doesn't show the exact function or process which consumes high energy.

Other solutions in this matter include, eLens which is accounted as a lightweight tool to estimate energy consumption and it is a code-based analysis model (31). In more detail, eLens checks the recorded information to perform analysis and returns energy-related results. EcoDroid is a software to rank the apps based on their energy usage and it uses both dynamic and static analysis (32). While dynamic models create fake test data, the static model gets the graph with the cost variable. It is important to mention that EcoDroid's accuracy is close to the Trepn. Similarly, the GreenOracle is using a trained model which uses big data to collect specific information such as system calls, and CPU utilization, to train and estimate power consumption (33). It also provides an opportunity to generate reports about energy consumption while changing the code base. In addition, Joulemeter (34), (35) is software that allows estimating energy usage of computer hardware utilization and consumption of software applications. It shows the resource usage for each component, and function, for instance, CPU utilization, and models the energy usage for them. In other words, we can also apply hardware configuration and Joulemeter will work in a machine-specific way.

Table 2.1 summarizes all tools mentioned in above sections including hardware-based measurement tools. *Unlike other previous work, we focus on developing a solution that can estimate energy consumption of applications just by taking a picture from a (thermal) camera.*

### 2.1.3 Energy Modelling and Estimation

Resource modelling for battery usage is a common method to estimate the energy consumption of hardware and software in general. The main approaches for modeling energy are mostly implemented within the software itself and/or in a middleware. By monitoring workload and computational effort of the resources running the application, it is then possible to extrapolate energy consumption metrics (36). Besides this, there are various estimation models and formulas that have been used for energy estimation

and measurement. A common solution that is adopted widely is the processes-level profiling tool called pTop, which uses a model which calculates the sum of the consumed energy for each process by using the following formula (37).

$$E_{appi} = \sum U_{ij} \times E_{resourcej} + E_{interaction} \tag{2.1}$$

where $U_{ij}$ illustrates consumption of process $i$ in the resource $j$. Besides the above formula, the general model was proposed to cover each resource. Basically, the model is a mathematical formula of reads/writes, and transitions in the resource itself.

$$E_{resourcej} = \sum_{jinS} P_j t_j + \sum_{kinT} n_k E_k \tag{2.2}$$

where $S$ is the state of the resource (read/write), $T$ is a transition, $P_j$ is consumed power at t interval, $E_k$ is the energy consumed during the transition, and $n_k$ is the number of transitions.

From the general formula (2.2), it is possible to generate functions for each resource such as CPU;

$$E_{CPU} = \sum_{j} P_j t_j + \sum_{k} n_k E_k \tag{2.3}$$

where $P_j$ is power consumption and $t_j$ is running time, $n_k$ is transition number and $E_k$ is energy consumed during that transition.

For Network Interface,

$$E_{Neti}^{f,v} = t_{sendi} \times P_{send} + t_{recvi} \times P_{recvi} \tag{2.4}$$

where $t_{sendi}$ and $t_{recv}$ are amount of time while process is sent and received respectively, $P_{send}$ and $P_{recv}$ are power usage during network interface receive and send operations.

For Hard disk,

$$E_{Diski} = t_{readi} \times P_{read} + t_{writei} \times P_{write} \tag{2.5}$$

where $t_{readi}$ and $t_{writei}$ are the spent time during read/write processes, $P_{read}$ and $P_{write}$ are the power consumption during read/write processes.

**Table 2.1:** Comparison of Energy Measurement Approaches

| Approach | Energy Measurement | Monitored Resources | Energy Precision | Hardware Support |
|---|---|---|---|---|
| Monsoon Powermeter | ✓ | Hardware Energy | Hardware | ✓ |
| LEAP | ✓ | Mobile Applications | Component level Hardware | ✓ |
| PowerScope | ✓ | Hardware and Processes | Process | ✓ |
| Energy Checker | ✓ | Hardware resources, Application Counters | Application | ✓ |
| Greenminer | ✓ | Hardware | Application level | ✓ |
| PowerTutor | × | Software Resources | Software, Classes and Methods | × |
| PCE | × | Application | Source Code | × |
| PowerAPI | ✓ | Hardware Resources | Process | × |
| JALEN | ✓ | Software resources | Code | × |
| JouleMeter | ✓ | Hardware | Process | × |
| Watts Up? Pro | ✓ | Hardware | Application Level | ✓ |
| Trepn | ✓ | Hardware Resources | Component level Hardware | × |
| eLens | ✓ | Code Base | Source code | × |
| EcoDroid | ✓ | Application | Source Code | × |
| GreenOracle | ✓ | Hardware and Processes | Process | × |

Also, PowerAPI is a tool to monitor consumption in real-time and illustrate the usage of CPU processes whilst Jalen is a software-level tool and providers hardware-related estimations. Generally, formula 2.2 shows the standard formula for the CPU model by using the Complementary Metal Oxide Semiconductor (38) equation as shown in the following formula.

$$Power_{CPU}^{f,v} = c \times f \times V^2 \tag{2.6}$$

where $f$ is the frequency of CPU, $V$ is voltage, and $c$ constant value based on the hardware material itself.

While theoretical models can be used to estimate energy consumption, they need to be tailored for a specific device and type of hardware. *Unlike others, we focus on linking*

*changes in thermal measurements with energy consumption of the device by looking at how the surface how the device heats on time due to the application processing.*

### 2.1.4 Limitations of Energy Measurement Techniques

As mentioned above, hardware-based tools require a careful and time-consuming setup since one wrong step can cause damage to the devices which is costly (39). Additionally, we cannot get information about system activities. Software-based tools are created by analyzing the hardware-based tools during building the model. To precisely represent the power consumption of the operations and components, the model should be built extremely carefully. However, software-based tools are too behind hardware-based tools from an accuracy perspective (40). Another limitation is the ground-truth values which are based on different tools and accuracies are different as well. This is because these tools are proprietary-based and vendors implement their own methodologies for energy consumption, which may introduce differences in the measurements. Nevertheless, differences may be marginal, and energy consumption patterns remain relative. Normalization of data is thus required for analysis of energy between tools/solutions.

## 2.2 Thermal Imaging

Thermal imaging uses the concept of measuring heat dissipation to estimate temperature. By looking at changes in light in the invisible spectrum using cameras, thermal imaging can determine heat of surfaces. Also, according to Kirk J. Havens, and Edward J. Sharp "Thermal imaging is simply the process of converting infrared (IR) radiation (heat) into visible images that depict the spatial distribution of temperature differences in a scene viewed by a thermal camera" (41). Undoubtedly, thermal cameras can perform tasks in the ambiance without light and also can pass through dense smoke and mist. Initially, specifically during the 1950s-1960s, thermal cameras evolved for military goals because it was essential for better surveillance applications. The first thermal camera was heavy, large, and expensive at that time. Also, they were cooled by the nitrogen liquid which was the requirement of cameras at the time. Eventually, it enabled a lot of tasks that were difficult to perform before while these cameras advanced over time. By the time, thermal cameras became available for civilians to apply in industrial applications, surveillance, robotics, etc (41).

If we want to dig into how thermal imaging works, it should be clear that all materials above 0 °K (≈ - 273 °C) emit infrared radiation. Infrared radiation is located in the electromagnetic spectrum between light and radio waves (42), (43) such that bare human eye can't recognize these lights because it is beyond visible light spectrum Table 2.2. Moreover, infrared radiation has a wide range of wavelengths between 0.7 $\mu m$ and 1000 $\mu m$ however common usage is from 0.7$\mu m$ to 20$\mu m$ for example usual temperature measurements (44). Generally, the infrared spectrum consists of three types of the range which are Near Infrared (NIR), Mid Infrared (MID), and Far Infrared (FIR). NIR spectrum has a wavelength range between 0.7$\mu m$ and 2.5$\mu m$ which can perform imaging operations on objects that have from 600 °C to 1000 °C temperature. MIR imaging has a wavelength between 1.3 $\mu m$ to 8 $\mu m$ and recognizes objects that have a temperature from 5 °C to 300 °C. The cameras that belong to the FIR spectrum uses wavelength between 7.5$\mu m$ and 13$\mu m$ with the capabilities of observing the temperatures between -20 °C and 900 °C and this is the most common spectrum for the cameras which are available commercially since they are smaller and cheaper (45). Eventually, thermal images illustrate the objects based on their temperature and are built with the help of heat signatures, and maps. In other words, the camera's thermal imaging sensor performs a transformation from the infrared energy to electrical signals to present in the form of a thermal image with colors (46). It is clear that the object in the open air is heating and becomes a heat source. In this case, thermal cameras have sensors inside to define the calibration period in order to create accurate thermal images. During the calibration process losing frames is inevitable because a new set of temperatures are inserted in front of the current frames.

Currently, there are a lot of thermal cameras manufactured, for example, Cat S60 smartphones are equipped with FLIR thermal cameras that have 80 x 60 infrared radiation sensors (The Figure 2.2 shows an example image produced by a Cat S60 Flir camera).

## 2.3 Applications of Thermal Imaging

As advancements occur in the thermal imaging field and thermal cameras, this section shortly presents its applications in the different fields. Thermal imaging has a wide range of applications areas, as most novel technologies, it has also been adopted by the

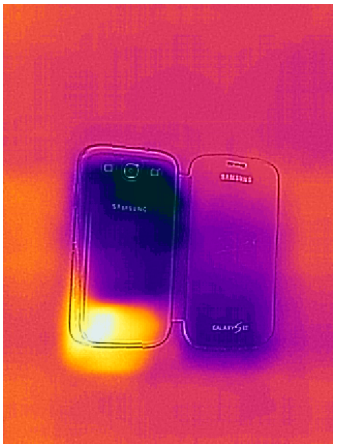


**Figure 2.2:** Thermal Dissipation from a Smartphone resulting from Application Usage

**Table 2.2:** Electromagnetic Spectrum(1)

| Wave | Wavelength (meters) | Frequency (Hz) |
|---|---|---|
| AM Radio | $10^{2}$ | $10^{6}$ |
| FM, TV | 1 | $10^{8}$ |
| Radar | $10^{-2}$ | $10^{9}$ |
| Infared | $10^{-6}$ | $10^{13}$ |
| Visible Light | $10^{-7}$ | $10^{15}$ |
| Ultraviolent (UV) | $10^{-8}$ | $10^{16}$ |
| X-Rays | $10^{-10}$ | $10^{18}$ |
| Gamma Rays | $10^{-14}$ | $10^{21}$ |

military as mentioned. A lot of researches had been conducted with the help of thermal imaging such as in agriculture, medicine, information technology, military, etc (47).

### 2.3.1 General Applications of Thermal Imaging

In medical analysis, CAD (Computer Aided Design) systems can help to monitor fever patients in places with a vast amount of people such as in borders, airports, and malls while the existence of different infections (maybe contagious) might be detected in this way. Especially last years, outbreaks of pandemic situations could be detected in crowded areas (48). Additionally, surveillance systems are one the most critical systems for defense and security, various researches have been investigated and (49) proposes 2 algorithms to embed inside the surveillance systems which can not only monitor but also perform detection tasks during poor lighting. The agricultural industry tries to take advantage of the recent technologies because of being a critical industry for economies. Visual inspection, which is subjective and labor-intensive, is commonly used to assess crop quality (50). *Unlike others, we develop a solution that can be used to estimate energy consumption from application running in smart, IoT and wearable devices.*

### 2.3.2 Application of Thermal Imaging in Energy Measurement

Since energy usage is getting a trending topic in the world, authors of (51) introduce a term that denotes the software's energy misbehavior called energy bugs by the authors. As expected, it decreases user satisfaction, increases the chance of battery drain, and so on. Authors of (11) introduce a method of energy usage analysis by taking energy anomalies into consideration such that the Carat application is developed which collects data from the smartphone and shows an overview of the consumption. A lot of anomaly cases were fed into the Carat application and Carat has been used by more than 500.000 IOS and Android devices. This is believed as the first collaborative diagnosis approach that is helpful to diagnose energy bugs. Approaches to estimate temperature changes depending on processing load have been proposed. For instance, there is a method of identifying heat sources directly from commercial processors (52). During the experiment, the authors used 2 Intel CPUs which resulted in high accuracy. Those results from their research illustrate the feasibility of the proposed method that is highly dependent on temperature data.

To supplement the temperature measurements from the embedded sensors, thermal models that can estimate the temperature profile of the entire chip or all thermally susceptible locations on the chip are required. For that purpose, interpolation-based

approaches for computing the full-chip thermal map from sensor values have been presented (53). Algorithms are designed to be clever to choose the optimal placement of sensors within a given budget because of the effect of the number and placement of embedded sensors on the accuracy of the results (54) (55).

Thermal solutions for smart, IoT and wearable devices have also been proposed. Mobile devices emit energy as they are active and executing any task, which denotes devices as heat sources. Thermal readings have been used in various researches in order to understand the movement of the heat on the components. The primary source of heat is CPU, GPU processes, and battery (56). In order to illustrate the temperature of each part of the device, a device that is called a Therminator is useful such that it is linked between layers (57). Additionally, in the Paterna et al. (58) the effect of ambient colors, variations, and component relations on the temperature is analyzed with the model. Even though a lot of research offers some understanding of the energy metrics in the mobile phone, it needs unexpected orchestration. However, we keep the processes simple by modeling energy using thermal images. So that, our experiments don't require any difficult mechanisms.

Additionally, thermal imaging is used by research (59) which targets a single integrated chip and adopts spatially resolved imaging of microprocessor power as a method. The core idea is that method considers the closeness of conduction and electricity as well (60). Authors of (61) create a method that pays attention to the multiple circuit components to evaluate and monitor with an IR camera. Similarly, thermal imaging has been used by authors of (4) in a way that inspired this research. The authors propose a novel approach to measuring energy consumption by using the energy footprints of smartphones, wearables, and other devices. Our research recalled the experimental setup guidelines from the methods that are proposed by the authors. *Unlike other work, we study thermal imaging further to estimate energy consumption of different types of applications.*

### 2.3.3 Advantages and Limitations of Thermal Imaging

Thermal cameras are highly useful in the cases where we need individuals' identification however for the identity of individuals, thermal cameras are not enough to do it compared to light cameras. Additionally, thermal cameras are well suited for most outdoor

environments over common visual cameras since they can observe temperature differences in objects quickly (62) such as fire, human activities, body temperature, etc (63). You can easily recognize objects to differentiate between animals, and humans, which is important for surveillance purposes. The reason is that thermal cameras are responsive to the radiation and produce the resulting image without any blur, or distortion even in inconvenient weather conditions. From the temperature estimations perspective, measuring the temperature of a large surface is more precise than point-to-point. However, it is not highly precise temperature measurement by the thermal cameras as contact methods.

Eventually, dependence on the shape, color, and some properties of the objects is one of the main limitations of the current thermal imaging techniques because it focuses on the reflections and absorption mainly. For instance, it is not possible to capture heat through specific materials like water, or glass because of being highly reflective in the thermal spectrum. This is one of the main disadvantages in some cases in which these objects are needed to be captured.

## 2.4 Summary

We presented a literature review on state-of-the-art thermal imaging (passive sensing) in addition to the thermal cameras which are used to detect thermal infrared radiation that is unrecognizable to the bare human eye. Even though thermal cameras have their own disadvantages, their positives exceed their drawbacks, especially, one that is their capability to see in dark and difficult environments. For instance, night vision systems work similarly to the eye in that they can detect and amplify small amounts of light. However, in completely dark places, night vision cameras are utterly worthless due to the lack of a light source. In contrast, thermal cameras can't perceive clear light. They can detect heat radiation and the objects' temperature fluctuations that are reflected back instead.

In the following Chapter, we conduct an analysis to confirm the feasibility of the approach to evaluate whether thermal imaging is useful to get an insight into the energy consumption of the phone by doing a couple of experiments.

# 3

# Motivation

In this Chapter, we present the motivation behind the work, the feasibility experimental testbed, underlying procedure and also the results for feasibility experiments. We also perform statistical testing to show that the thermal imaging can indeed be used to model and estimate the energy consumption of smartphones, IoT devices and other wearable devices.

## 3.1 Preliminary Rationale

The energy measurement and estimation solutions used for measuring and estimating energy mentioned in the state-of-the-art solutions are tedious to use. The hardware-based methods being precise have a limitation in trying to measure energy as modern devices such as smartphones have a non-detachable battery. The software methods on the other hand are less precise and do not give an actual estimate of the energy. We begin this Chapter by demonstrating the feasibility of an alternate solution to measure and estimate the energy of smartphones based on thermal imaging. The primary motivation for this research can be illustrated by considering a real-life scenario where a person previously used a power monitor to check the power consumption of a phone. But at the moment, he does not have a hardware device or he purchased a new phone with a non-detachable battery that does not allow for the use of hardware. Nevertheless, he can get insights into energy consumption by looking at the phone's thermal absorption. The current technological trend is to produce phones without detachable batteries (64) and many wearables and IoT devices, which disables hardware-based mea-

surement techniques. Because the heating pattern of the phone is caused by processes, threads, CPU usage, and sensor usage, it may be possible to gain some insight into energy consumption by analyzing the thermal footprint of the device when it is being used. This approach could benefit not only smartphones and smartphone users, but also small IoT devices such as wearables. However, the most important aspect of the research was the feasibility analysis, which describes the feasibility of the study and allows for further analysis.

## 3.2 Feasibility Analysis

We now demonstrate that thermal imaging can be a promising solution for measuring and estimating the energy of smartphone devices. We ran a few applications on a smartphone and calculated the energy consumption of the three individual applications by mapping thermal dissipation energy to energy estimation. We also analyze thermal images from smartphones running different applications and extract the thermal values and also compare the results with the energy measurement values from the power monitor.

### 3.2.1 Experimental Testbed

We conduct several experiments with three different energy-consuming and different resource-intensive applications and also analyze the thermal absorption for these three different applications at specific intervals. The applications chosen were $YouTube$, $Chess$, and $Google\,Chrome$. These applications were run on Samsung Galaxy S3 running on the Android-based OS version. We use a Thermal Logger application running on CAT S60 Smartphone. The Thermal Logger app takes the thermal image and also generates the temperature values and emissivity values of the respective thermal images. It also has the functionality to set the FPS value based on our needs. To analyze the thermal footprints, we set the FPS value as 1. For recording the thermal video and eventually extracting the thermal images from the video, We use a tripod stand to obtain clear thermal footage. The tripod is kept at a distance of 40 cm from the android phone running the applications.

### 3.2.2 Procedure

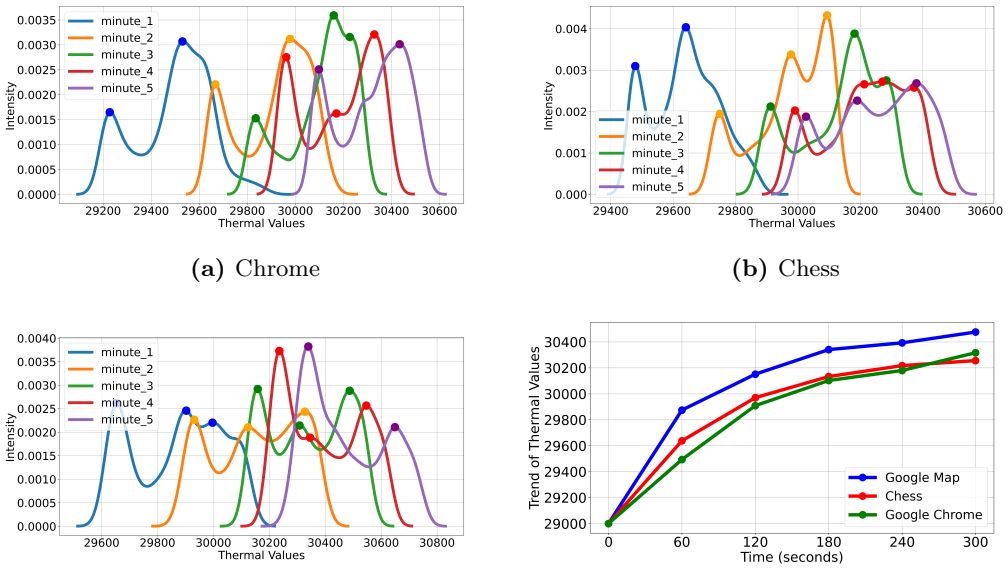
Before the applications are run on Samsung Galaxy S3 running the three different applications, the device was kept in the refrigerator for 10 minutes to cool down. Then the device was taken out and put on an experimental table. The table was covered by a black sheet to obtain the clean thermal footage. Before running the applications the phone was kept on the table for five minutes to let it adapt to the ambient room temperature of 25 degrees Celsius. One application out of the three applications was run on the Samsung Galaxy S3 and the Thermal Logger application was also started on the CAT phone fixed on a tripod stand. The thermal image along with the temperature and emissivity values associated with the thermal image was extracted from the Thermal Logger application. The same procedure was followed for two other applications. The application was run for a period of 5 minutes and each application was run for three iterations. Then the data from the experiment was extracted which include the thermal image frames from the thermal footage along with other temperature and emissivity values. The extracted data were collected for further processing to measure and estimate the energy consumption of these three different applications.

### 3.2.3 Data Preprocessing and Analysis

After conducting the experiments with the three applications mentioned above, we collect the thermal frames from the footage, the respective emissivity values, and thermal thermal values from each iteration of usage of the application. The data is subjected to data cleaning and processing steps which are explained in detail in Algorithm 1. As demonstrated in the algorithm, the thermal images involve resizing in order to remove noise from the surroundings. Then the thermal values at each specific interval are extracted after applying the image processing techniques mentioned in the algorithm. These thermal values are analyzed to generate the plots which are demonstrated in the results subsection below.

### 3.2.4 Results

Figures 3.1a, 3.1b, and 3.1c show the resulting plot for each application that we have used for experiments. The results show thermal values and their corresponding intensity values for each time interval between 1 to 5 minutes. From the graph, it is clear that




**(a)** Chrome

**(b)** Chess

**(c)** Google Map

**(d)** Mean of Thermal Values

**Figure 3.1:** Feasibility Experiment Results: a)-c) Thermal Intensity Graphs of Three Applications; d) Mean Thermal Value Trend for each Application

as time progresses we have higher thermal values as is evident from the thermal values. The higher time intervals also show more intensity peaks due to the heating up of the device from prolonged application usage.

We also analyzed the data to find the behavior of the mean of extracted thermal values from the thermal arrays over time-specific and equal time intervals ranging between $0-300$ second interval. The intervals were chosen at the end of every 60 second. Figure 3.1d shows the results. It is clear from the graph that the device gets heated while running different applications over time. The mean value of the thermal values curve for specific applications shows a different pattern at the specific time intervals highlighted in the graph. For instance, the Google Chrome application at the end of the 240 second interval overtakes the chess application as is evident from the figure. The mean value of curves for all applications also tends to achieve a state of stability as is demonstrated at the end of the 180 second interval.

### 3.2.5 Statistical Testing

Additionally, it was crucial to use statistical tests to examine our data mathematically in order to determine whether or not it had a distribution, thus we conducted both

**Table 3.1:** Statistical Analysis

| Test | Statistic | P-value | Interpretation |
|---|---|---|---|
| Shapiro-Wilk | 0.952 | 0 | Does not look Gaussian |
| D'Agostino and Pearson's | 296.638 | 0 | Does not look Gaussian |
| Mann-Whitney U | 3359400.500 | 0 | Different distribution |
| Kruskal-Wallis H | 8340.077 | 0 | Different distribution |

parametric and non-parametric testing. Non-parametric tests do not require that the data be distributed normally, whereas parametric tests examine if the data follows a normal distribution (65). Distribution-free tests are another name for non-parametric tests. Each extracted thermal array was subjected to the Shapiro-Wilk, D'Agostino, Pearson's, and Anderson-Darling tests in order to validate the data.

The test findings made it obvious that our data is not normally distributed, as shown in Table 3.1. This expression was supported by the results of the Mann-Whitney U and Kruskal-Wallis H tests, which indicated that the data was distributed differently. Therefore, it was more worthwhile to look at the link between thermal values rather than the distribution of the data in order to identify some correlation.

### 3.2.6 Insights from Results

The main insights that can be inferred from the results of the feasibility experiments are highlighted below:

- As time progresses when any application is used, we see higher thermal values generated from power and energy consumption.
- The higher time interval thermal intensity curve shows more peaks in thermal values than smaller time intervals.
- Some applications start with low heat and end up overtaking other applications.
- The thermal values start to go to a peak value and then tend to achieve stability.

## 3.3 Summary

Overall, this Chapter covered the primary motivations for the investigation as well as a feasibility study conducted before the major experiment. The results of the feasibility analysis were therefore close and visually correlated, demonstrating that it is practical and worthwhile to proceed forward.

# 4

# Methodology and Pipeline

Our work deals with estimating energy from thermal footprint by leveraging thermal imaging to generate thermal footprint from devices such as smartphones and other IoT devices. Our feasibility experiments show that the thermal footprint of devices can be used to estimate energy consumption. In this Chapter, we will describe the complete methodology for the measurement and estimation of energy from the thermal footprint. We will also shed some light on the thermal dissipation footprint background and also describe the pipeline by explaining the main algorithms and methods.

## 4.1 Thermal Dissipation Background

The working principle of Thermal Imaging from the spectrum's perspective has been stated in Chapter 2. In addition to that thermal imaging shows us the radiated heat of an object by comparing temperature with the objects around because thermal cameras let us record the temperature of the different objects.

Basic physics dictates that all objects with a temperature greater than $0K$ emit radiation (66). The components of smartphones and other IoT devices, such as the CPU, battery, network card, etc., also cause them to produce this heat. This emission is sometimes referred to as thermal radiation or blackbody radiation, depending on the object's temperature, which determines the radiation power. Also known as blackbodies, perfect emitters are things that have the greatest radiation power. The temperature can theoretically be calculated using the blackbody law. Another way to think of a blackbody is as an idealized object that can both absorb and emit energy at all

wavelengths. Planck's law serves as an example of the blackbody radiation theory. Using the following formula and Planck's law, we can calculate the power of emitted radiation by using wavelength ($lambda$) and temperature ($T$) (67).

$$I(\lambda, T) = (\frac{8\pi c}{\lambda^5})(e^{\frac{hc}{k\lambda T}} - 1)^{-4} \tag{4.1}$$

In formula 4.1 $h$ is Planck's constant ($6.626176 \times 10^{-34} Js$), $c$ is the speed of light ($3 \times 10^8\ m/s$), and $k$ is Boltzmann's constant ($1.38 \times 10^{-23}\ J/K$). In this matter, it is good to mention Stefan-Boltzmann's law which shows the relevance of total energy flow over all wavelengths in the specified range. As the components are generating heat, the heat is radiated through the case and outer parts of the smartphone as illustrated in the following Stefan-Boltzmann law.

$$E = \epsilon \sigma T^4 \tag{4.2}$$

In formula 4.2, $\sigma$ is the Stefan-Boltzmann constant ($5.670367 \times 10^{-8} Wm^{-2}K^{-4}$), $E$ is the total amount of radiation, and $T$ is the temperature. It is clear from the Formulae 4.1 and 4.2 both power and energy have a direct proportional behaviour with temperature and this is the reason we will also be using temperature (thermal values) along with the thermal images in order to model and estimate energy consumption of devices.

## 4.2 Processing and Estimation Pipeline

We will present the complete processing and estimation modeling pipeline for energy consumption. The pipeline begins with extracting data using the Thermal Logger application installed on the CAT S60 Smartphone and ends with mapping the thermal values to the energy consumption of a device doing some computational task. The complete pipeline is shown in Figure 4.1. This section explains the primary thermal pipeline steps, including all the components, procedures, algorithms, and mathematical concepts utilized in this study for estimating energy consumption from thermal imaging.

### 4.2.1 Getting Input Data

We set the Thermal Logger Application on the CAT S60 smartphone to record the thermal frames from the device whose energy is to be measured. Its implementation details are described in Section 5. The Thermal Logger application is implemented in

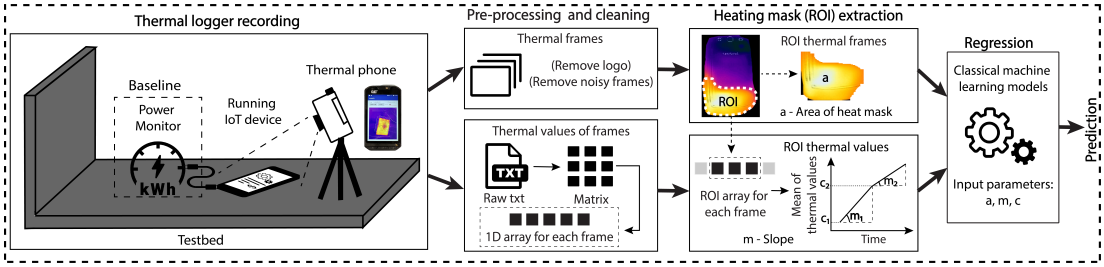



**Figure 4.1:** Proposed Methodology and Pipeline

such a way that it is incorporated to leverage the FLIR thermal camera of the CAT smartphone. Once the application is run, it gives us the option to set the Frames per Second (FPS). The FPS decides how many frames to extract per second. The Thermal Logger application generates the thermal image and its corresponding RGB image in the visible spectrum, emissivity values in text format, and also a $2D$ array of thermal values in the form of a text file. All these filenames are also included with a timestamp at which the particular frame was captured which is useful in analyzing and estimating thermal footprint.

### 4.2.2 Data Pre-processing

The process of transforming data into the form where we can actually manipulate and process it is the most crucial part of data analysis. Prior to pre-processing, it is essential to first make a list of all the problems with the input data.

- The thermal image dimensions are twice as large as the thermal values array dimensions due to the raw data format from Thermal Logger application.
- The array of thermal values is a text file, which makes it unsuitable for mathematical operations.
- For every specific $n^{th}$ time interval, we have a large number of thermal frames to process depending on the set FPS value of the Thermal Logger Application.

The first stage in the pre-processing was determining the appropriate number of frames to start our processing with. We did this by converting the timestamp into DateTime format, which made it easier for us to choose the desired number of thermal frames and their respective emissivity and thermal values. Once the desired number of

frames to process are identified, we scale these thermal images to eliminate the noise and make them identical to the dimensions of $2D$ thermal arrays. We converted the text files containing $2D$ thermal arrays into Python's Numpy arrays. After this, we have NumPy arrays for each specific time interval together with the application name that is running on the device and corresponding thermal image paths. This facilitates the analysis and visualization which we will revisit in a later subsection.

### 4.2.3 Thermal Image Processing

We now have pre-processed data from the text files of the Thermal Logger application that is ready for further analysis. We next start pre-processing of thermal image frames to extract the region of interest (ROIs) which denotes the main regions on the device that are getting heated. In order to extract the Region of Interest, the thermal images are subjected to image processing steps and techniques mentioned in Algorithm 1. We capture these ROIs using the built-in and potent functionalities of Python's OpenCV Package[1]. For each application, the image processing function loops through the file directories. To capture the contours of the thermal fingerprint, we chose to use the built-in and potent functionalities of Python's OpenCV package. The initial step in thermal image processing captures ROIs as contours. The thermal image is converted into grayscale with the help of `cv2.COLOR_BGR2GRAY` available in OpenCV. These grayscale images are further passed into `cv2.threshold` function with the different values for each application and for specified time intervals. Thresholding preserves values over the threshold value while filtering out pixel values below it, producing a cropped image. These threshold values are sent as a parameter to the `cv2.findcountours` function, which returns the coordinates of the heated area in the image. Since we need the thermal values rather than the pixel values, the processed thermal images are saved into new folders and the extracted coordinates from the contours are used to capture the same area from the thermal thermal arrays as well.

Secondly, we created one more function as shown in Algorithm 1 to process the contours extracted from the thermal images again to remove cold pixels and keep only the heat mask. We performed morphological transformation by using the $9x9$ kernel window and the `cv2.morphologyEx` function's `MORPH_CLOSE` and `MORPH_OPEN` parameters to construct masks. Since bitwise operators are incredibly helpful for these kinds

[1] https://opencv.org/

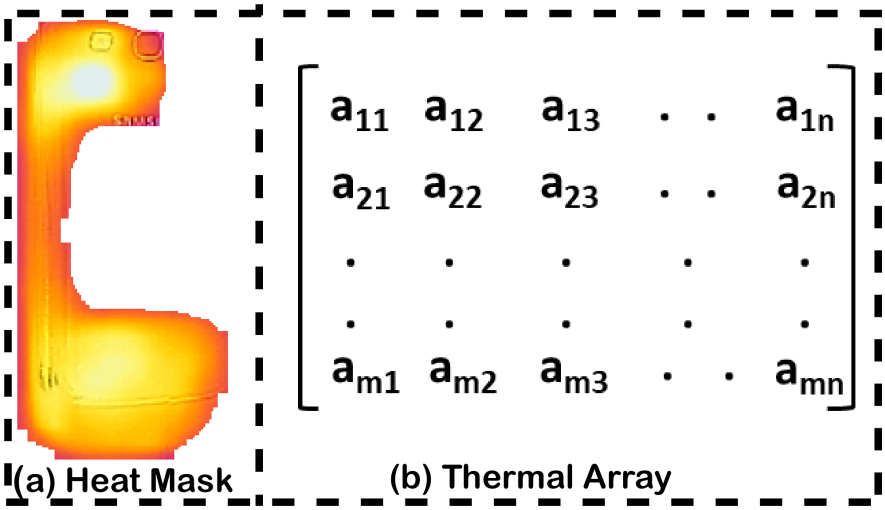



**Figure 4.2:** Sample Output from Algorithm 1

of masking tasks, we use them for masking. By removing the pixels that are below the threshold values and have an irregular shape as is the case in a thermal footprint, the bitwise `AND` operation eventually gave us the sole portion of the image containing the heated part as shown in Figure 4.2. Now since we have both the ROI (extracted thermal footprint) and the corresponding thermal array, these will be fed to the analysis part to gain better insights.

Overall, this step is really crucial and results in extracted heat footprint image as illustrated in Figure 4.2, and an array of thermal values (Figure 4.2) of that footprint area. This will be fed into the data analysis part in order to process and analyze data to get better insights.

### 4.2.4 Processing Results

We now have proper data that can be processed to create graphs, charts, diagrams, and tables. The extracted arrays of thermal values contain 0 and they need to be dropped therefore we flattened the 2D arrays into 1D and dropped all 0 values. Python's Pandas module is used here to store data in a tabular format and to perform data analysis leveraging robust Pandas algorithms. Pandas dataframe with the following headings is created: $AppName$, $Time$, $MeanThermalValue$. This stage concludes with the resulting Pandas dataframe, which contains important information for the next step.

**Algorithm 1:** Image processing - calculating ROI, contours, heat mask

**Input:** List of applications with corresponding paths to thermal images and thermal values array files

**Output:** Exported heat mask (ROI) image and corresponding slice of thermal array

```
   /* Contours Extraction                                                    */
 1 for each app in apps do
 2     image ← Conversion to Grayscale;
 3     image ← Apply Threshold;
 4     image ← Detect all contours;
 5     initialize heatContoursExtractedResult ;  /* initialize dictionary to
         store paths to extracted images, and to store extracted thermal
         arrays  */
 6     for each contour in c do
 7         x, y, w, h ← cv2.boundingRect() ;     /* Get contour coordinates */
 8         ROI ← image[y:y+h, x:x+w];
 9         thermalValues ← thermal_array[y:y+h, x:x+w];
10         Append thermalValues to heatContoursExtractedResult;
11         Save ROI as an image in the new path;
12         Append new ROI path to heatContoursExtractedResult;
13     end
14 end
   /* Mask Extraction                                                        */
15 for each app in apps do
16     image ← Conversion to Grayscale, Apply Threshold;
17     kernel ← 9x9Kernelwindow mask ← cv2.morphologyEx| with
         cv2.MORPH_CLOSE| and cv2.MORPH_OPEN| parameter;
18     result ← Conversion to BGRA and 3rd channel is mask ;
19     result ← cv2.bitwise_and(result, result, mask)| ;
20     heatCountoursExtractedResult ← 0 for cell where the coordinate
         is 0 in the result ;
21     end
22
```

### 4.2.5 Plotting and Visualizations

Plots are essential for data analysis because they provide visual interpretation. In this case, it is critical to use Python's plotting tools to build graphs, which we did with Seaborn and Matplotlib. These libraries include sophisticated tools for manipulating and generating various types of graphs. The graphs below help to comprehend the following:

- We use `seaborn.distplot()` method to generate a thermal intensity distribution graph from the thermal values in our thermal array (except 0s).
- We use `matplotlib` to generate a graph to see the trend of the mean of thermal values of each time interval for each application (`plot()` function).

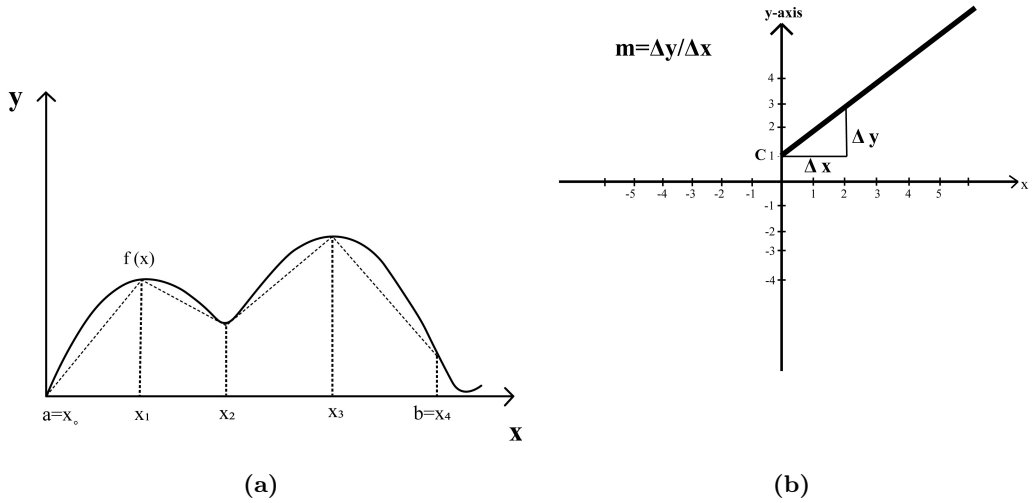
Overall, this section deals with the generation of various visualization plots that will be used in the modeling of energy estimation in the next subsection.

### 4.2.6 Modelling and Estimation

Now we have the plots and other visualization graphs that give us an insight into how the thermal values behave with different application usage. We also have the intensity of thermal values at each particular time interval. We calculate the area under the thermal intensity curve that denotes the amount of heat absorbed by the device whose energy is to be estimated. We also have the trend of thermal values over time. The behavior of the thermal values over time intervals can be explained with the help of the general equation of a straight line. We extract the slope ($m$) and the intercept at each time interval ($c$) to explain the behavior of the curve and hence the energy estimation and modeling.

**Mathematical Background:** We now explain the mathematical background behind the modeling of the plots that we generated into estimating the thermal energy of the device. We first present the mathematical background behind finding the area under the thermal intensity curve and also the trend of thermal values at specific intervals of time.

**The Trapezoidal Rule:** The thermal intensity curve can be assumed a function $f(x)$ continuous over time interval $[a, b]$. The numerical integration of this function over the




**(a)** **(b)**

**Figure 4.3:** Graphical Representation: (a) Trapezoidal Rule for Calculating **Area a**, (b) Linear Function Graph having a **Slope m** and **Intercept c**

definite interval $[a, b]$, gives us the approximation of the area under the curve.

$$Area = \int_a^b f(x)\,\mathrm{d}x \tag{4.3}$$

Let $n$ be any positive integer and $\Delta(x) = \frac{b-a}{n}$. Let us divide the time interval $[a, b]$ into $n$ sub-intervals, each with length $x$, and place their endpoints at $P = x_0, x_1, x_2..., x_n$ as shown in Figure 4.3a. The Trapezoidal rule for calculating the area under the curve is calculated using the following equation:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{2}\Delta(x)[(f(x_0) + 2f(x_1) + 2f(x_2) + ... + 2f(x_{n-1}) + f(x_n)] \tag{4.4}$$

We use this equation to calculate the area under the intensity curve. The area under this thermal curve (temperature values) gives us an idea about the amount of energy heat absorbed by the device (68).

**Formula of Straight line:** A car driving at a constant speed across an ever-increasing distance is just one example of many situations where there is continual change over time. In this instance, the motion of a car is described by a linear function with a constant rate of change. The rate of the change can be both positive or negative and is given by the slope of the change at any given time interval. There are several ways to

express linear functions, such as tabular form, function form, graphical form, and others. A linear function, which is a mathematical formula, represents a straight line having some intercept which is denoted by $c$. The most common linear function equation is denoted by:

$$y = mx + c \tag{4.5}$$

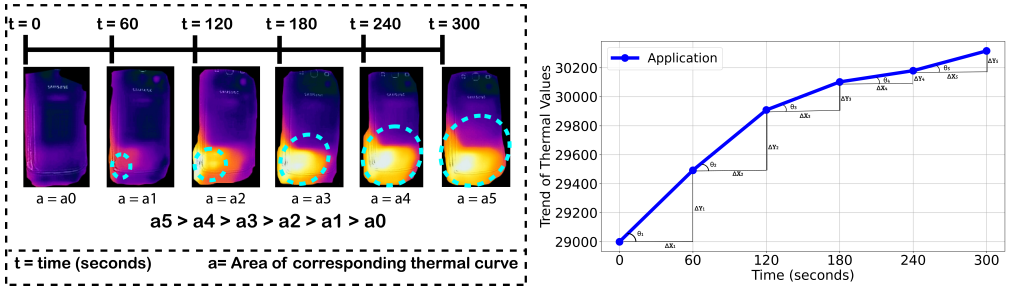
In the above equation; $m$ is the slope (increment or decrement) at any given time instant, $c$ is the intercept (beginning value) of the line, $x$ is the independent variable along the $X$-axis and $y$ is the dependent variable along $Y$-axis.

The equation of the straight line can be interpreted in the context of the rate of change which we will also use in explaining the usage behavior of different applications. This rate of change is denoted by the slope ($m$) in the equation. In the previous car example, the distance traveled grows over time since it is a function of time at a constant speed and starting distance. The rate of change that gauges how quickly dependent variables change in proportion to the independent variable is referred to in this context as speed.

$$m = \frac{\text{vertical change}}{\text{horizontal change}} = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \tag{4.6}$$

Furthermore, it is vital to describe the graphical representation of the equation of the straight line because our research focuses on numerous graphs to estimate the energy consumption from different devices. Figure 4.3b denotes the graphical representation of the equation of the straight line. The steepness of the line in the graph shown in the figure demonstrates whether the function increases, decreases or remains constant.

However, there are times when we must determine the slope and intercept values on our own. Mathematically, the slope, which is denoted by a $m$ variable, indicates how steep the line is. We must first determine the increment of the vertical difference over the increment of the horizontal difference between the two locations in order to calculate it from the graph. Given two possible values $x_1$, $x_2$ where the difference $x_1$-$x_2$ denotes horizontal change and can be written as $\Delta$x while $y_1$, $y_2$ whose difference denotes vertical change and can be written as $\Delta$y, also can be described in the form ($x_1$, $x_2$) and ($y_1$, $y_2$) as a set of points, can be calculated as illustrated in the Formula 4.6. Recalling




**(a)** **(b)**

**Figure 4.4:** Estimating Energy from Estimation Parameters: (a) Trend of **Area a** over Time, (b) Trend of Thermal Values for a Particular Application with Illustrated Linear Lines having a **Slope m** and **Intercept c**

Figure 4.3b, also defines the slope of the line between two data points, eventually, in the sense of interpretation, a greater slope (m), a steeper line, and a greater increment rate and vice-versa (69).

**Explaining the Energy Consumption Behaviour of Applications from Area $a$, Slope $m$ and Intercept $c$:** We have demonstrated the mathematical background of the trapezoidal rule, linear function with some slope $m$ and intercept $c$ as shown in equations 4.3, 4.4, 4.5 and 4.6. Now let us demonstrate the application of these equations in estimating the energy consumption of devices and also explain the behavior of the application usage in devices. We will also discuss how to interpret the graphs that show the increase in the mean of thermal values over a specific time. The area under the thermal intensity curve as shown in Figure 4.3a gives us the idea of how much the device is getting heated up. As mentioned earlier too, the area under the thermal intensity curve gives us an idea of how much heat is absorbed by the device, and eventually, the energy requirement is high. Figure 4.4a depicts how area of actual heated parts increases with prolonged usage of device.

For demonstrating the usage of the mentioned equations in estimating energy consumption using thermal imaging, we will use the thermal behavioural trend of applications from Section 3. We have also put the similar demo figure here as Figure 4.4b. As shown in the figure, the thermal values of the application for each time interval have a different slope and intercept. The slope and intercept of the lines for each time interval give us an idea of how the energy consumption behaved during that particular time interval. The slope denotes the rate at which the energy consumption is taking place and

the intercept of the line for a particular time interval gives us the value of the estimated energy. From the figure, we have four different linear functions that correspond to each time interval of 60 seconds. In order to find the slope in our use case, $\Delta$y is the rate of change of thermal values, $\Delta$x is the rate of change of time, which is equal to 60 seconds in the figure. The ratio of $\Delta$y by $\Delta$x will give us the slope of the four different linear functions. We can also calculate the intercept (c) of each of the four lines that give us the thermal values at that time interval. In other words, it is possible to understand the applications' behavior with the help of m and c values because it tells us, how much energy the application consumes at first, and how fast it increases and ends. On the graph, it is clear that some applications can use high energy when it starts however later on increases gradually. On the other hand, some applications show low intercept, but the slope is high which leads to surpassing other applications as time progresses.

**Building a Regression Model:** The parameters $Area(a)$, $Slope(m)$, and $Intercept(c)$ are used to build a regression model that maps and estimates the results of our solution to the *Power* measured using Monsoon Power Monitor as demonstrated in Section 5. We use Bayesian ARD regression, Ridge Regression (Tikhonov regularization), and Bayesian ridge regression from from $scikit-learn$ as estimators to estimate the energy consumption.

## 4.3 Summary

This Chapter describes the methodology of the proposed solution that can be used to estimate the energy consumption of different devices. We presented the complete pipeline that involves processing, visualization, and mathematical modeling of the behavior of application usage by leveraging the area, slope, and intercept of the thermal value curve over a specified time interval.

# 5

# Experimental Setup

We evaluate our proposed solution for estimating energy using thermal imaging by performing rigorous experiments to demonstrate how the proposed solution estimates energy from devices in use. We perform two sets of experiments to evaluate the performance of our solution. The first set of experiments are performed on smartphones and the second set of experiments are performed on IoT devices. We also compare our results with baseline results from the Monsoon power monitor. In the following subsections, we describe the experimental testbeds and procedures.

## 5.1 Smartphone Testbed

We first demonstrate the experimental testbed and procedure for evaluating the performance of estimating energy from thermal imaging on a smartphone. We select five applications to be run on smartphones and also run them at fixed time intervals to evaluate the energy using thermal imaging.

### 5.1.1 Apparatus

We used an off-the-shelf Caterpillar (CAT) S60 smartphone equipped with a FLIR thermal camera. We also use a Thermal logger application running on this CAT S60 smartphone and rely on FLIR thermal camera to capture the thermal imaging frames and their associated thermal emissivity values and thermal values. The number of thermal images and their respective emissivity and thermal values are specified by the

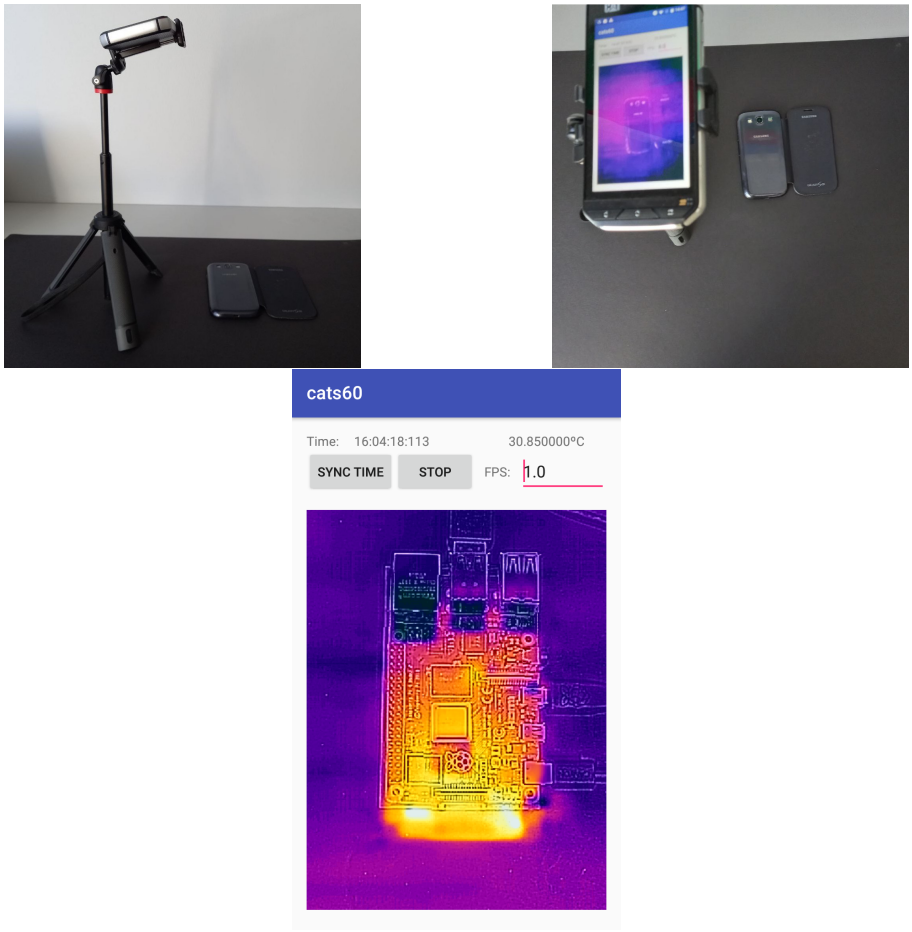



**Figure 5.1:** Experimental Testbed involving Samsung Galaxy S3 Smartphone

FPS value on the Thermal Logger application. The applications are executed on the Samsung Galaxy S3 smartphone running on an Android-based operating system.

### 5.1.2 Applications Selected for Evaluation and Estimation of Energy

To begin, it is critical to select the appropriate applications for the experiment because there are numerous factors that influence energy consumption. In this section, we will discuss why we chose specific applications, as well as general statistics and information about them. Understanding what affects energy consumption is crucial. The application source is unquestionably more significant than the application category. The category of the application can have an effect, but the history and evolution of the application are what really matter, according to this research (70). There are 34 categories in the

Google Play store (8 games and 26 applications), but we didn't exactly adhere to the categories.

The fact that code smell can affect energy utilization is undeniable, though, and this fact compelled us to pick applications carefully, especially those that had received positive feedback from the audience (71). To put it briefly, the source code is crucial in this aspect. It is important to note that because battery level has little impact on energy usage (72), we won't take it into account for this experiment. As previously mentioned, source code is essential for this reason since CPU and memory utilization directly increases energy consumption (31). To put these findings in nutshell, the main issues governing the energy consumption for application usage are listed below:

- Unnecessary background operations - It is important to conduct an experiment by being careful with background operations.
- CPU activities
- High RAM utilization
- Running extra sensors (for instance, GPS),Graphical rendering (for high-resolution games)

From the above insights, it follows that it would be excellent to categorize apps on our own by taking into account the sort of energy utilization of that program, rather than based on their Play Store category. We need to know how many background processes are active for the program, as well as its approximate CPU and memory usage, whether it makes use of additional sensors, and whether it engages in high-graphical rendering, advertising, synchronization, and other activities. Overall, the insights show that the structure of the source code, not the number of lines of code, is more important for energy usage. To prevent code smells, we will employ well-known programs, and a variety of applications, some of which involve sensors, the internet, and graphical rendering.

We chose the following applications after taking all of those factors into account, including the phone's capabilities and Android version:

- Video Player - The main job of this application is video rendering. It uses integrated graphics, RAM and also processor.

- Google Maps[1] - Map - It uses HTTP request along with other hardware resources like GPS sensor with real-time synchronization.
- Flight Simulator[2] - It is a game and uses High graphical rendering and calibration sensors.
- Facebook Messenger[3] - It is an instant chat application.
- AR Solar System[4] - This application is an AR/VR based application and uses High graphical rendering.

### 5.1.3 Thermal Logger Application

This application collects different log files and uses FLIR Thermal camera integrated into the Caterpillar S60 smartphone. This application collects data from the thermal camera which involve thermal images and the respective RGB image, emissivity values, and also the thermal values along with the timestamps. The application also has an option to set the FPS value based on our needs. Once the FPS value is set, the application collects data based on the set FPS Value.

**Implementation:** We implement the Thermal Logger application in Android Studio. The application is designed to operate in four phases. These are *MonitorService*, *LoggerService*, *StorageService* and *LogAppend* phase. The *MonitorService* is responsible for initializing and terminating the Thermal Logger application and also for synchronizing the time-stamps. The *LoggerService* phase is responsible for connecting the service to the thermal camera sensor that the CAT S60 Smartphone is using. It is also responsible for receiving thermal frames, the corresponding thermal and emissivity values, and also the RGB image from the thermal camera as per the fps value which is set before starting the logger application. The *StorageService* deals with getting storage access on the device to record all the logs that have been collected in the earlier phase. Finally, the last phase *LogAppend* appends logs based on timestamp values. Once the data collection is done, the *MonitorService* ensures that the session is terminated and data and log collection are terminated.

[1]https://play.google.com/store/apps/details?id=com.google.android.apps.maps
[2]https://play.google.com/store/apps/details?id=com.fungames.flightpilot
[3]https://play.google.com/store/apps/details?id=com.facebook.orca
[4]https://play.google.com/store/apps/details?id=com.ar.solar

**Setting the FPS Value:** FPS denotes the number of frames per second in a video. The higher the FPS value, the higher the quality and smoothness of the video. The FPS setting was critical in our experimental setup experiment to ensure the optimal quality without having to analyze a lot of files. As a result, we conducted several experiments to compare the results from different FPS values. We gathered results from these different FPS values in order to compare image quality and other information like emissivity values and thermal values. Eventually, we concluded that the FPS of 8 frames per second is optimal to conduct experiments and also ensure a sufficient amount of frames and other related log files are captured. The optimal value of FPS makes sense as the FPS value of the FLIR thermal camera integrated into the CAT S60 smartphone is also around 8 frames per second.

### 5.1.4 Procedure

There are a few tweaks, but the procedure is the same as it was for the feasibility study experiment (4). We run the five different applications on Samsung Galaxy S3. Before running the application, we put the phone inside a refrigerator for five minutes to obtain a baseline temperature. After that, we take the phone from the refrigerator and run the applications one by one. Each application was run three times to account for inaccurate data and we then average the results. The CAT S60 smartphone is mounted on a tripod stand at a distance of 30 *cm* from the phone running these applications. The Thermal Logger application running on the CAT S60 Smartphone records the thermal images and their respective emissivity and thermal values from the phone running these applications. Each application is run for a fixed time interval of five minutes. The recorded data is then subjected to additional pre-processing and analysis. The experimental testbed is shown in Figure 5.2.

## 5.2 IoT Device Testbed

To evaluate our proposed solution for estimating the energy of IoT devices, we run the edge-only deployment mode of DeFog Benchmark suite (73) on a Raspberry Pi 4B (RPi4). We deploy three applications on the Raspberry Pi with varying workloads to estimate the energy footprint of the IoT device.


**Figure 5.2:** Experiment Setup Flow

### 5.2.1 Apparatus

We use the same apparatus that we used for estimating the energy footprint of the smartphone. The only difference is that instead of Samsung Galaxy S3 running five different applications, we used a Raspberry Pi 4B having a Quad-core Cortex-A72 (ARM v8) processor with a RAM of 4 GB. Once the application is running on the raspberry, the Thermal Logger application installed on a CAT S60 Smartphone mounted on a tripod stand captures the thermal frames with emissivity and thermal values along with the timestamps. This data is further analyzed and processed to estimate energy.

### 5.2.2 Applications Selected for Evaluation and Estimation of Energy

We use three different applications from DeFog Benchmark suite - YOLOv3, PocketSphinx, and Aeneas. These applications have been selected as they require varying bandwidth and have different latencies. From a computational perspective, these applications offer different workloads on the device and are thus ideally suited for our experiments.

- YOLOv3 - It is an object classification-based application that uses deep learning along with a pre-trained model and weights. It is a bandwidth-intensive and computational-intensive application. A single neural network is used throughout the entire image by the YOLO object detection system. The image is divided into

various regions, and each region's bounding boxes and probabilities are calculated. The projected probabilities are used to weight these bounding boxes. A pre-trained model is used by the original application to resize a supplied image asset and identify objects in the image. With percentage weights, a labeled image is created that provides the accuracy of the estimation.

- PocketSphinx - It is a speech-to-text conversion application. It uses a trained acoustic model to convert speech into text. It is also a bandwidth-intensive and computational-intensive application. In order to determine the source and destination languages for speech-to-text conversion, a provided audio file (.wav) is converted to a specified language in text form using a pre-trained acoustic model. The integrated resources come from a large speech repository. The end user's device sends an a.wav file to the edge container of the application. To speed up text-to-speech conversion, an acoustic model that has already been trained is offloaded to the IoT device running the application.

- Aeneas - It is a text-audio forced alignment application. It uses an audio file and a text file to map text to audio. It is a bandwidth-intensive application. Forced alignment is the process of creating synchronization maps for a collection of text snippets and an audio file that contains the pertinent material. For each text asset that is submitted, Aeneas determines the mapping between the associated audio segment. The end-user device sends an audio file (.wav) to the edge node. On the device, text segmentation takes place. The text segment (.xhtml) is supplied as an asset to the edge for this application, where forced alignment takes place. To make it more responsive for the end-user, the task at the edge is calculating the synchronization map between the text fragment and audio fragment.

### 5.2.3 Procedure

The procedure here follows a similar pattern. We run the applications on the IoT device one by one. The applications are deployed on the IoT device by using the edge deployment mode of DeFog Benchmark Suite. The applications are deployed on the IoT device - Raspberry Pi 4B using Docker containers. The applications are deployed one by one. We run one application three times starting with a 1 user workload. Then the workload is increased to 5 concurrent users and eventually to 10 concurrent users.

The Thermal Logger application installed on the CAT S60 smartphone mounted on a tripod stand captures the thermal image and other files like emissivity and thermal values. The time interval is not fixed here. We pass the assets mentioned in the above section for a particular application involving a single user or multiple concurrent users and wait for the output to be generated.

## 5.3 Baseline Experiments using Monsoon Power Monitor

We also conducted baseline experiments using Monsoon Power Monitor. These baseline experiments serve as a method of validating our current solution for estimating energy consumption from thermal imaging. In this section, we discuss the apparatus and procedure for measuring energy using a Monsoon Power Monitor.

### 5.3.1 Apparatus

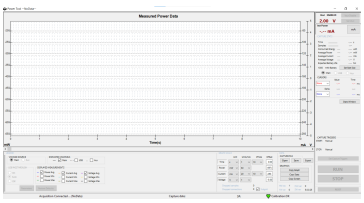
The Monsoon Power Monitor device is our primary tool for conducting baseline experiments, and in order to control it and visualize the results, PowerTool(Figure 5.3b) software must be downloaded. Additionally, it enables the analysis of power usage up to a 3-cell battery (15). The devices that are monitored for measuring energy include the Samsung Galaxy S3 android phone and Raspberry Pi 4B that we used for performing experiments.

### 5.3.2 Procedure

For measuring the energy consumption for an experimental setup involving a Samsung Galaxy S3 Smartphone, we configured the Monsoon Power monitor by following the official documentation and user manual[1]. The input voltage was set to 4 Volt as is required for the phone. We ran the applications one by one on the phone and recorded the energy consumption by each application for a specified interval of 5 minutes. The same procedure was followed for measuring the energy consumption of Raspberry Pi. The input voltage was set to 5 Volt. We ran the applications one by one by following the procedure mentioned in the experimental setup for IoT. The energy measurement was collected from the PowerTool software that was installed on the laptop. The recorded

[1]https://www.msoon.com/hvpm-product-documentation

**(a)** Testbed



**(b)** Power Tool Software User Interface


**(c)** Experiment Flow

**Figure 5.3:** Power Monitor (Baseline) Testbed with Experimental Flow

energy values at each instant were saved into a comma-separated (.csv) file for further analysis and insights. Figure 5.3c summarizes the experimental flow and procedure.

## 5.4 Summary

We described the experimental testbed for estimating energy using thermal imaging for two sets of devices - one for a smartphone running an Android-based OS and the other for an IoT device (Raspberry Pi 4B). We also described the experimental flow for conducting baseline experiments using a Monsoon Power Monitor for both sets of experiments that serves as the basis for validation of our proposed research.

# 6

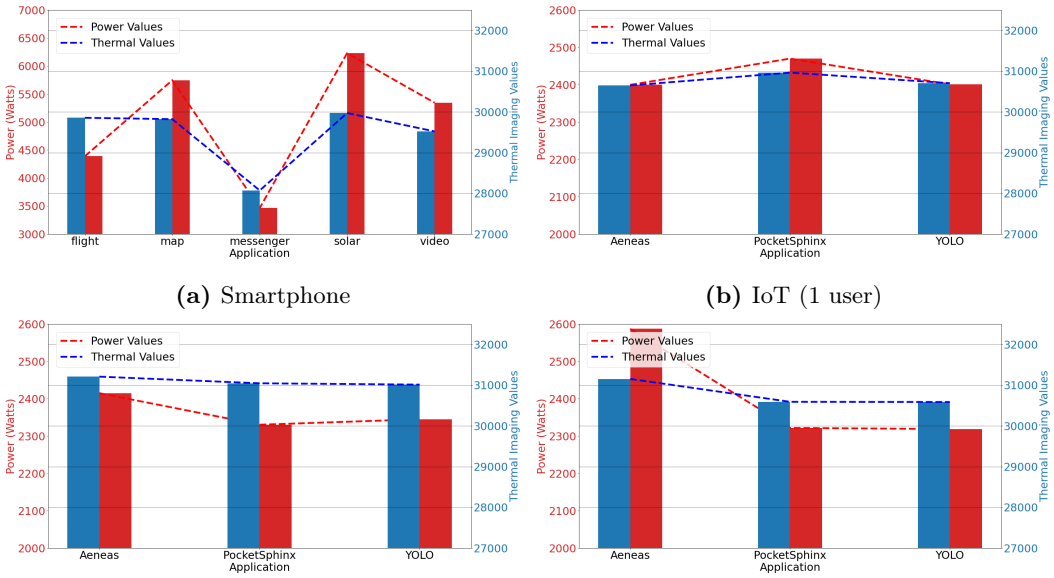
# Results

This Chapter demonstrates the results and other insights from experiments that were conducted on the smartphone (Samsung Galaxy S3) and the IoT device (Raspberry Pi 4B). We will first present the baseline results from Monsoon Power Monitor and also compare these baseline results to validate our experimental results.

## 6.1 Highlights from Results

The proposed method of estimation of energy from thermal imaging deals with finding three parameters - $area$, $slope(m)$, and $intercept(c)$. The main highlights of the results are:

- The results are compared with the baseline results from Power Monitor. The KS-Test shows that the energy consumption trend is similar in both cases.
- The estimation of energy involves the extraction of three parameters that collectively give us an idea about the energy consumption of a device. These parameters are area, slope, and intercept.
- The area under the thermal intensity curve increases as the time for which the application is run on the device increases. This gives us an idea of how much heat is absorbed by the device in the form of a thermal footprint.
- The peaks in the thermal intensity curve indicate how much computationally intensive an application is.
- The slope of the trend of thermal values of particular application usage over specific time intervals is either positive, negative, or sometimes zero. The slope gives us the




**(a)** Smartphone

**(b)** IoT (1 user)

**(c)** IoT (5 concurrent users)

**(d)** IoT (10 concurrent users)

**Figure 6.1:** Comparison with Baseline for Smartphone and IoT with Varying Userload

rate at which the device is heating up. In other words, we can say that it gives us the rate at which energy consumption takes place.

- The intercept of the thermal values over an increase in time interval increases. This gives us information about the actual thermal value at that particular time interval.
- The performance of Regression Model to predict the estimated energy consumption with the true energy from Monsoon Power Monitor shows $RMSE$ of around 0.10 with $R^2$ around 0.86.

## 6.2 Comparison with Results from Monsoon Power Monitor (Baseline)

We have used Monsoon Power Monitor as our baseline. We compare the results and other findings from our main experimental results to those of power values being measured in Watts from Power Monitor. A power monitor experiment is critical to have a baseline to understand and confirm our findings. First, we compare the results of power values measured from the smartphone by the Monsoon power monitor to that of values from thermal imaging. Figure 6.1a shows the comparison of results.

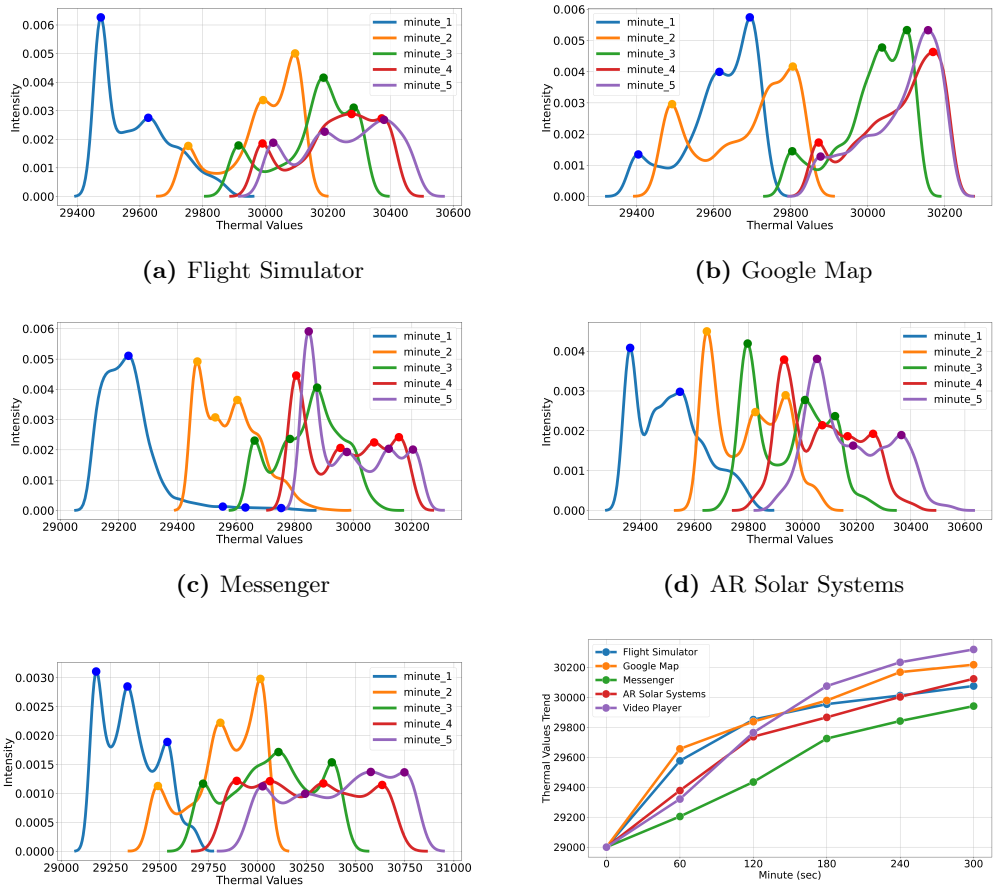
We also compare the values of the thermal heat of Raspberry with that of power monitor values. Figures 6.1b, 6.1c, and 6.1d show the results comparison of baseline measurement values from the power monitor to that of thermal values. As shown from the results, the baseline and results from thermal imaging measurements follow the same trend. As expected, the Messenger application is the lowest since it is just an instant chat application while AR Solar System is the highest power consumption by using high graphical rendering and being an AR-based application. The Kolmogorov-Smirnov (KS) test to compare the relative differences in thermal imaging results and power monitor verify that data is similar in both cases. The KS test results are presented in Table 6.1.

**Table 6.1:** KS test and Kendall Correlation for Comparing the Results with Baseline

| Device Type | KS Statistic, D | p-value | Kendall $\tau$ |
|---|---|---|---|
| Smartphone Applications | 0.2 | 1 (>0.05) | 0.7 |
| Raspberry Pi (1 user) | 0.33 | 1 (>0.05) | 1 |
| Raspberry Pi (5 concurrent users) | 0.33 | 1 (>0.05) | 1 |
| Raspberry Pi (10 concurrent users) | 0.33 | 1 (>0.05) | 1 |

## 6.3 Performance Evaluation on Smartphone

This section describes the evaluation of our proposed approach to estimating the energy from smartphones using thermal imaging. Figures 6.2a, 6.2b, 6.2c, 6.2d, 6.2e show the results of the intensity of thermal values for five different applications. As seen from the results, for each application, the intensity peaks increase with an increase in a time interval. From the figures, it can be seen that the peak intensity points of thermal values increase as we increase the time period of application usage on the smartphone from 1 minute to 5 minutes. These peaks correspond to local minima and local maxima points as highlighted in the figure. The reason that these peak points increase with an increase in the time duration of application usage is that with time the smartphone gets heated with continuous usage and we see a surge in peak points. Also figure illustrates that curve shifts to the right as time progresses because thermal values are getting increased as the smartphone gets heated. Additionally, we summarized the area under




**(a)** Flight Simulator

**(b)** Google Map

**(c)** Messenger

**(d)** AR Solar Systems

**(e)** Video Player

**(f)** Trend of Thermal Value

**Figure 6.2:** Experiment Results for Smartphone (Intensity and Trend graphs)

intensity curves along with peak point coordinates in Table 6.2. This area under the intensity curve shows the amount of heat absorbed by the smartphone. As seen from the table, the area under the curves increases gradually from 1 minute duration to 5 minute duration.

We also plot the thermal values after specified time intervals as shown in Figure 6.2f. As we can see from the figure, the trend gradually increases when the time interval for each application usage increases. The applications at the end of the time interval of 60 seconds show messenger as the low energy-consuming application, followed by video player, AR Solar System, Flight Simulator, and then Google Maps. At the end of the 5 minute interval, the video player overtakes all the other applications by consuming

**Table 6.2:** Estimation Parameters for Smartphone: $m$-Slope; $\Delta$m-Change in Slope; $c$-Intercept; $\Delta$c-Change in Intercept; $a$-Area; $\Delta$a-Change in Area

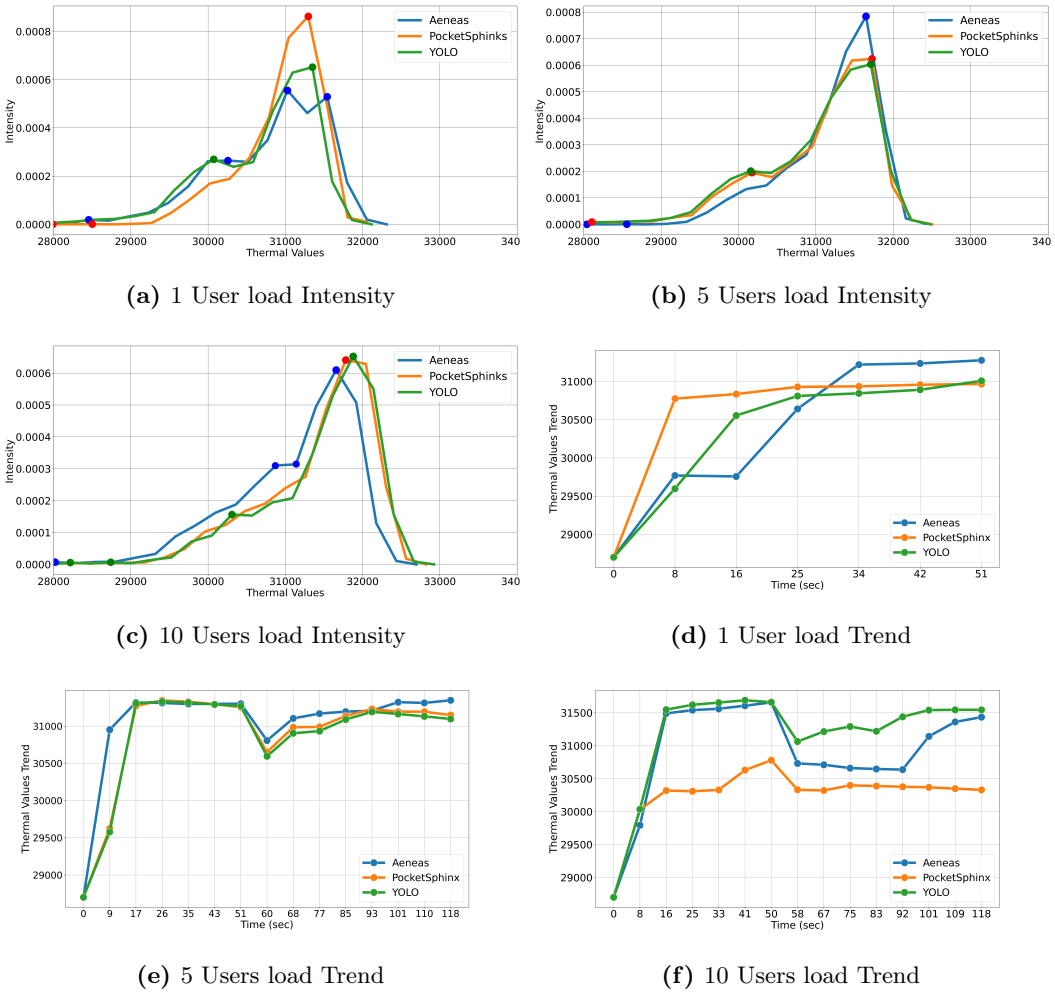
| Time | m | Δm | c | Δc | a | Δa |
|---|---|---|---|---|---|---|
| **Flight Simulator** | | | | | | |
| **0-60** | 6.40 | 6.40 | 29000 | 29000 | 0.9063 | 0.9063 |
| **60-120** | 7.79 | 1.39 | 29383.70 | 383.70 | 0.9069 | 0.0006 |
| **120-180** | 1.72 | −6.07 | 29850.91 | 467.21 | 0.9074 | 0.0005 |
| **180-240** | 0.96 | −0.76 | 29954.38 | 103.47 | 0.9075 | 0.0001 |
| **240-300** | 1.06 | 0.1 | 30011.95 | 57.57 | 0.9080 | 0.0005 |
| **Google Map** | | | | | | |
| **0-60** | 10.93 | 10.93 | 29000 | 29000 | 0.9070 | 0.9070 |
| **60-120** | 3.03 | −7.9 | 29655.99 | 655.99 | 0.9084 | 0.0014 |
| **120-180** | 2.35 | −0.68 | 29837.77 | 181.78 | 0.9086 | 0.0002 |
| **180-240** | 3.16 | 0.81 | 29978.62 | 140.85 | 0.9090 | 0.0004 |
| **240-300** | 0.83 | −2.33 | 30168.01 | 189.39 | 0.9090 | 0 |
| **Messenger** | | | | | | |
| **0-60** | 3.41 | 3.41 | 29000 | 29000 | 0.9065 | 0.9065 |
| **60-120** | 3.84 | 0.43 | 29204.53 | 204.53 | 0.9070 | 0.0005 |
| **120-180** | 4.83 | 0.99 | 29434.71 | 230.18 | 0.9085 | 0.0015 |
| **180-240** | 1.95 | −2.88 | 29724.67 | 289.96 | 0.9089 | 0.0004 |
| **240-300** | 1.66 | −0.29 | 29841.68 | 117.01 | 0.9089 | 0 |
| **AR Solar Systems** | | | | | | |
| **0-60** | 6.29 | 6.29 | 29000 | 29000 | 0.9078 | 0.9078 |
| **60-120** | 5.99 | −0.3 | 29377.67 | 377.67 | 0.9096 | 0.0028 |
| **120-180** | 2.15 | −3.84 | 29737.26 | 359.59 | 0.9096 | 0 |
| **180-240** | 2.28 | 0.13 | 29866.18 | 128.92 | 0.9097 | 0.0001 |
| **240-300** | 2.01 | −0.27 | 30002.91 | 136.73 | 0.9097 | 0 |
| **Video Player** | | | | | | |
| **0-60** | 5.35 | 5.35 | 29000 | 29000 | 0.9068 | 0.9068 |
| **60-120** | 7.39 | 2.04 | 29320.89 | 320.89 | 0.9068 | 0 |
| **120-180** | 5.17 | −2.22 | 29764.57 | 443.68 | 0.9075 | 0.0007 |
| **180-240** | 2.64 | −2.53 | 30075.03 | 310.46 | 0.9086 | 0.0016 |
| **240-300** | 1.43 | −1.21 | 30233.64 | 158.61 | 0.9088 | 0 |

more energy. The behavioral trend that the application shows is explained with the help of intercept $c$ and slope $m$ in the form of Table 6.2. As seen from the Table, slope $m$ gives us an idea of the rate at which the application usage on smartphone heats up the phone and eventually rate of energy consumption. The intercept $c$ gives us an idea of the temperature of smartphone at any given time interval. As seen from the table, the slope $m$ increases abruptly once the application is loaded and it is for this reason that both slope $m$ at the end of $0 - 60$ second period is the highest. Then the slope either is positive or negative depending on the application type usage. The intercept value $c$ on the other hand goes on increasing with an increase in time.

## 6.4 Performance Evaluation on IoT Device

We next evaluate the performance of our proposed solution by attempting to estimate the energy consumption of the IoT device, Raspberry Pi 4B. We use three applications - YOLO, PocketSphinx, and Aeneas as mentioned in the experimental setup (Section 5). Figures 6.3a, 6.3b, and 6.3c show the results of thermal heat intensity peaks. The results show three applications run as a single user, 5 concurrent users, and 10 concurrent user requests. The intensity values tend to go higher when we increase the number of concurrent requests for each application. The peak points which correspond to spikes in heat intensity also show an increasing trend as we move from single user to multiple concurrent user workloads. Some intensity peaks are smaller while some are large. Table 6.3 summarizes statistics about thermal intensity graphs. We also demonstrate that the area under the intensity curves corresponds to the amount of heat absorbed by the device and eventually more energy requirement increases for each application when we increase the workload.

The trend of thermal values is shown in Figures 6.3d, 6.3e, and 6.3f. The figure shows the trend of each of the three applications for 1 user request, 5 concurrent user requests, and 10 concurrent user requests separately. From the figures, the visual inspection shows YOLO and Aeneas as high energy consuming and PocketSphnix being the least power consuming. This trend in thermal values is due to the fact that YOLO and Aeneas are computationally and bandwidth-intensive as compared to PocketSphinx. The trend in thermal values is better explained in the form of the intercept $c$ and slope $m$ values as shown in Table 6.3. From the table, the slope $m$ denotes the rate at which




**(a)** 1 User load Intensity

**(b)** 5 Users load Intensity

**(c)** 10 Users load Intensity

**(d)** 1 User load Trend

**(e)** 5 Users load Trend

**(f)** 10 Users load Trend

**Figure 6.3:** Experiment Results for IoT Devices (Intensity and Trend graphs)

**Table 6.3:** Estimation Parameters for Raspberry Pi 4B: $m$-Slope; $\Delta$m-Change in Slope; $c$-Intercept; $\Delta$c-Change in Intercept; $a$-Area; $\Delta$a-Change in Area

| User-load | m | Δm | c | Δc | a | Δa |
|---|---|---|---|---|---|---|
| **Aeneas** | | | | | | |
| 1 | 42.85 | 42.85 | 30070.73 | 30070.73 | 0.842 | 0.842 |
| 5 | 18.90 | −23.95 | 31020.61 | 949.88 | 0.848 | 0.006 |
| 10 | 17.09 | −1.81 | 31981.88 | 961.27 | 0.852 | 0.004 |
| **YOLO** | | | | | | |
| 1 | 38.49 | 38.49 | 30083.15 | 30083.15 | 0.816 | 0.827 |
| 5 | 17.11 | −21.38 | 30842.62 | 759.47 | 0.820 | 0.006 |
| 10 | 18.34 | 1.23 | 31296.93 | 454.31 | 0.838 | 0.006 |
| **PocketSphinx** | | | | | | |
| 1 | 37.78 | 37.78 | 30371.99 | 30371.99 | 0.827 | 0.816 |
| 5 | 17.49 | −20.29 | 30871.29 | 499.3 | 0.833 | 0.004 |
| 10 | 13.64 | −3.85 | 31537.6 | 666.31 | 0.839 | 0.018 |

the thermal heat increases, and the intercept $c$ shows the actual value of thermal heat. We can see that the slope is sometimes positive or negative. The positive value denotes that the device is getting heated up and the negative value denotes that the device is not getting heated as compared to the time interval where we have a positive slope.

## 6.5 Energy Estimation

We now demonstrate the results of building a regression model that estimates the energy consumption from different IoT devices based on the extracted parameters from thermal imaging. These parameters are area, slope, and intercept. The baseline energy measurements from the Monsoon power monitor have been used as a label. We use regression models which estimate the energy consumption from the parameters extracted to predict power based on baseline Monsoon power monitor values. We use the common regression models like Bayesian ARD regression, Ridge Regression, and Bayesian ridge regression from the $scikit-learn$ library in python to predict the energy consumption using 10-fold cross-validation across the training and test data set. The results of R-Squared ($R^2$) and root mean square error (RMSE) are shown in Table 6.4.

| Parameter | Bayesian ARD | | Bayesian Ridge | | Ridge | |
|---|---|---|---|---|---|---|
| | **R-Squared** | **RMSE** | **R-Squared** | **RMSE** | **R-Squared** | **RMSE** |
| **(a) → EE** | 0.81 | 0.14 | 0.80 | 0.13 | 0.81 | 0.13 |
| **(a, m, c) → EE** | 0.86 | 0.10 | 0.87 | 0.10 | 0.86 | 0.10 |

**Table 6.4:** R-Squared and Root Mean Square Error (RMSE) of Estimating Energy (EE) for Different Regression Models. Model Data → Predicted. Regression Methods: Bayesian Automatic Relevance Determination Regression (ARD), Bayesian Ridge Regression and Ridge Regression). Features: Area (a), Slope (m) and Intercept (c).

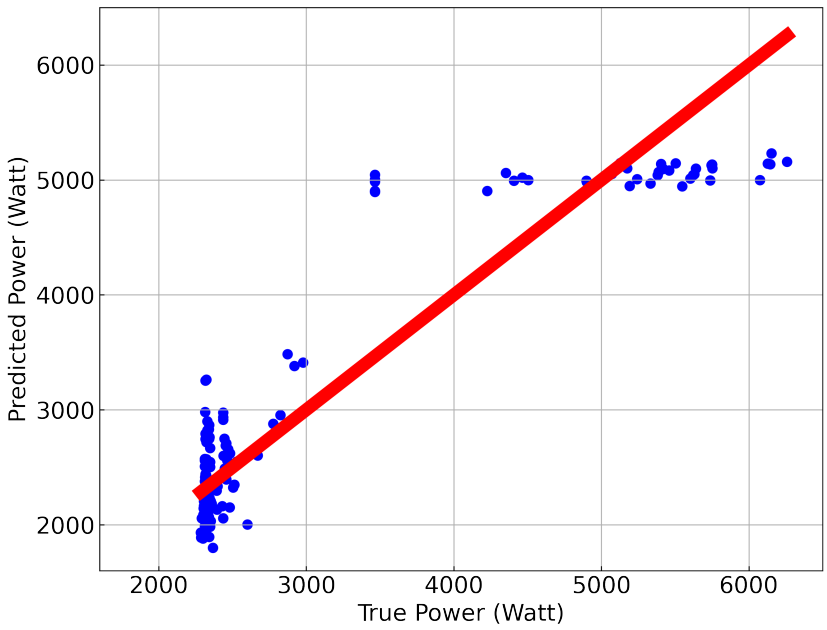



**Figure 6.4:** Estimation of Power from Regression Model

- As shown in Table, the RMSE and R-Squared of regression increase when we include the parameters $a$, $m$, and $c$. But it is clear that *area* is the main parameter that is important for estimating energy. The results are also visualized in Figure 6.4 where we plot the predicted power from the regression model with true power obtained from the baseline Monsoon power monitor results.

## 6.6 Summary

This Chapter presented the findings of our experiments. We demonstrated that thermal imaging can be used to estimate energy consumption from applications running in smartphones as well as estimating energy of IoT devices. Our results also demonstrate that the thermal imaging is also capable to differentiate energy consumption between different applications. In addition, our results also suggests that thermal imaging can estimate energy consumption of devices when handling different processing load (when handling requests from different users)

# 7

# Discussions

We have so far shown that thermal imaging can be used to estimate energy from devices like smartphones and other IoT devices. Now we will briefly discuss the potential application scenarios, the scope of extensions in other related fields, and also the main limitations.

## 7.1 Rooms for Improvement

We conducted our experimental testbeds by choosing two sets of devices, one Samsung Galaxy 3 and a Raspberry Pi $4B$. We ran a few applications on each of the devices with varying workloads and functionalities. The energy estimation can further be improved by conducting experiments with more sets of devices and also by increasing the number of applications to be tested. Furthermore, the segmentation techniques for the extraction of heat masks can be improved further with the help of some power image processing techniques. We have also used one Trapezoidal Rule for calculating the area under the thermal intensity curve. The use of some other area functions can also be explored. Last but not least, environmental conditions like room temperature and luminosity can also have an effect on thermal cameras.

## 7.2 Complementing Software Profilers

As we have already discussed in Chapter 2 that software profilers that are being used for measuring energy suffer from limitations that arise from debugging and other code-related run time issues. These issues can be mitigated greatly by using the software

profilers as a baseline and complementing its measurement values with energy estimation from our measurement approach. The issues related to debugging, code errors, and other runtime issues could be easily detected by measuring energy via thermal imaging.

## 7.3 Remote Monitoring

Our solution has the potential to be incorporated into unmanned Aerial Vehicles (UAVs), which can enable batch monitoring of the IoTs being used for agriculture (74), forestry (75), and wildlife monitoring (76), etc. It is undeniable that in large fields, monitoring the devices, heat, and energy is an extremely time-consuming and resource-intensive task, so mounted thermal cameras on UAVs can decrease the complexity of the task (77).

## 7.4 Practicability

Measuring the energy while holding the device can induce noise in the thermal camera and hence in the pre-processing of the resulting thermal frame used for analysis. Thermal cameras capture the frames based on heat difference compared to the target's environment, which can be affected by human body temperature. Another practical point is that the system can be integrated into the manufacturing processes by mounting to the robot arm or any place to hold the thermal camera while the remote server is executing algorithms and producing results for the end user (e.g., monitoring officer). However it requires future research for special use cases and business requirements.

## 7.5 Industrial Use Cases

Our solution has the potential to be used for many industrial purposes. One use case can be in the manufacturing process of smartphones and other devices by taking thermal images continuously during the manufacturing process. Monitoring for energy consumption during fabrication and manufacturing can result in highly optimized fabrication and assemblage by ensuring Joule's Law of Heating is minimized to a great extent. Besides, it can also be used to constantly monitor old machines and other electrical wirings in factories and industries. Our solution can also be used to avoid electrical and other plumbing issues in industries in advance by constantly monitoring them (78).

## 7.6 Other Potential Application Scenarios

Our solution has the potential to be used in measuring energy consumption from modern portable devices which have miniaturized designs and do not have a detachable battery. It can also be used by mounting a thermal camera on top of IoTs. Besides estimation of energy, other potential application includes the estimation of the health of batteries (79) by monitoring for irregular heat patterns. Another application is that it can be used for monitoring hand-held devices at airports and other security devices to check for potential exploding devices. The same can also be used for monitoring and detection of explosives and other dangerous devices (80). National Security Agencies could greatly benefit from the use cases of this solution. This solution can be widely used by incorporating it with Wireless Sensor Networks (WSNs) for their constant management and monitoring. The field of Biometrics could also be greatly benefited from this work as our work can be used for liveliness detection in bio-metric sensors for detecting impersonated fingers, faces, Iris, and hands (81).

# 8

# Summary and Conclusion

We developed a novel technique for quantification and estimation of energy consumption for Smartphones and other IoT devices alike. We demonstrated the feasibility of this approach and also performed rigorous experiments involving different applications with varying usage requirements and user-load. We extract three different parameters from the thermal behavior of the devices that we use for the estimation and measurement of energy. These estimation parameters are area, slope and intercept. We also developed a regression model that maps the thermal values from thermal imaging and predicts the true power obtained from a hardware-based measurement device (Monsoon Power Monitor) with $RMSE$ of 0.10.

# Bibliography


[1] R. Vadivambal, D. S. Jayas, Applications of thermal imaging in agriculture and food industry—a review, Food and Bioprocess Technology 4 (2) (2011) 186–199. xi, 15

[2] H. Anwar, D. Pfahl, S. N. Srirama, Evaluating the impact of code smell refactoring on the energy consumption of android applications, in: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2019, pp. 82–86. doi:10.1109/SEAA.2019.00021. 3

[3] M. Gottschalk, J. Jelschen, A. Winter, Energy-efficient code by refactoring, Softwaretechnik-Trends: Vol. 33, No. 2. 3

[4] H. Flores, J. Hamberg, X. Li, T. Malmivirta, A. Zuniga, E. Lagerspetz, P. Nurmi, Evaluating energy-efficiency using thermal imaging, in: Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications, ACM, 2019, pp. 147–152. 3, 17, 41

[5] M. Proske, J. Winzer, M. Marwede, N. F. Nissen, K.-D. Lang, Obsolescence of electronics-the example of smartphones, in: 2016 Electronics Goes Green 2016+(EGG), IEEE, 2016, pp. 1–8. 3

[6] M. U. Khan, S. Abbas, S. U.-J. Lee, A. Abbas, Measuring power consumption in mobile devices for energy sustainable app development: A comparative study and challenges, Sustainable Computing: Informatics and Systems 31 (2021) 100589. 3

[7] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, A. De Lucia, Software-based energy profiling of android apps: Simple, efficient and reliable?, in: 2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER), IEEE, 2017, pp. 103–114. 3

[8] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shojafar, A. I. A. Ahmed, S. A. Madani, K. Saleem, J. J. Rodrigues, A survey on energy estimation and power modeling schemes for smartphone applications, International Journal of Communication Systems 30 (11) (2017) e3234. 3

[9] P. K. D. Pramanik, N. Sinhababu, B. Mukherjee, S. Padmanaban, A. Maity, B. K. Upadhyaya, J. B. Holm-Nielsen, P. Choudhury, Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage, IEEE Access 7 (2019) 182113–182172. 3

[10] Z. Ou, S. Dong, J. Dong, J. K. Nurminen, A. Ylä-Jääski, R. Wang, Characterize energy impact of concurrent network-intensive applications on mobile platforms, in: Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture, 2013, pp. 23–28. 3

[11] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, S. Tarkoma, Carat: Collaborative energy diagnosis for mobile devices, in: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys '13, Association for Computing Machinery, New York, NY, USA, 2013. doi:10.1145/2517351.2517354.
URL https://doi.org/10.1145/2517351.2517354 3, 16

[12] G. C. Holst, Common sense approach to thermal imaging, Vol. 1, SPIE Optical Engineering Press Washington, 2000. 4

[13] G. Pinto, F. Castor, Energy efficiency: A new concern for application software developers, Commun. ACM 60 (12) (2017) 68–75. doi:10.1145/3154384.
URL https://doi.org/10.1145/3154384 7

[14] J. Mancebo, F. García, C. Calero, A process for analysing the energy efficiency of software, Information and Software Technology 134 (2021) 106560. doi:https://doi.org/10.1016/j.infsof.2021.106560.
URL https://www.sciencedirect.com/science/article/pii/S0950584921000446 8

[15] I. Monsoon Solutions, Mobile Device Power Monitor Manual, Monsoon Solutions, Inc. 8, 44

# BIBLIOGRAPHY


[16] P. Peterson, D. Singh, W. Kaiser, P. Reiher, Investigating energy and security trade-offs in the classroom with the atom leap testbed, in: Investigating Energy and Security Trade-offs in the Classroom With the Atom LEAP Testbed, 2011, pp. 1–9. 8

[17] J. Flinn, M. Satyanarayanan, Powerscope: a tool for profiling the energy usage of mobile applications, in: Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 2–10. doi:10.1109/MCSA.1999.749272. 8

[18] Intel, Intel Energy Checker SDK.
URL http://software.intel.com/enus/articles/intel-energy-checker-sdk. 8

[19] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, S. Romansky, Greenminer: A hardware based mining software repositories software energy consumption framework, in: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 12–21. doi:10.1145/2597073.2597097.
URL https://doi.org/10.1145/2597073.2597097 9

[20] A. Hindle, Green mining: A methodology of relating software change and configuration to power consumption, Empirical Softw. Engg. 20 (2) (2015) 374–409. doi:10.1007/s10664-013-9276-6.
URL https://doi.org/10.1007/s10664-013-9276-6 9

[21] G. Procaccianti, P. Lago, W. Diesveld, Energy efficiency of orm approaches: An empirical evaluation, in: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1–10. doi:10.1145/2961111.2962586.
URL https://doi.org/10.1145/2961111.2962586 9

[22] Watts up? pro,.
URL https://www.wattsupmeters.com/secure/products.php?pn=0&wai=837&spec=4 9

[23] Arduino uno board,.
URL https://www.arduino.cc/en/main/arduinoBoardUno 9

[24] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, Y.-M. Wang, Fine-grained power modeling for smartphones using system call tracing, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 153–168. doi:10.1145/1966445.1966460.
URL https://doi.org/10.1145/1966445.1966460 9

[25] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, S. Tarkoma, Modeling, profiling, and debugging the energy consumption of mobile devices, ACM Comput. Surv. 48 (3). 9

[26] H. A. Le, A. T. Bui, N.-T. Truong, An approach to modeling and estimating power consumption of mobile applications, Mob. Netw. Appl. 24 (1) (2019) 124–133. doi:10.1007/s11036-018-1138-4.
URL https://doi.org/10.1007/s11036-018-1138-4 9

[27] A. Bourdon, A. Noureddine, R. Rouvoy, L. Seinturier, PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level, ERCIM News 92 (2013) 43–44.
URL https://hal.inria.fr/hal-00772454 9

[28] A. Noureddine, A. Bourdon, R. Rouvoy, L. Seinturier, A preliminary study of the impact of software engineering on greenit, in: 2012 First International Workshop on Green and Sustainable Software (GREENS), 2012, pp. 21–27. doi:10.1109/GREENS.2012.6224251. 9

[29] A. Noureddine, A. Bourdon, R. Rouvoy, L. Seinturier, Runtime monitoring of software energy hotspotsdoi:10.1145/2351676.2351699. 9

[30] Q. Technologies, Trepn Profiler Starter Edition: User Guide.
URL https://developer.qualcomm.com/qfile/28788/trepn_6.0s_starter_edition_user_guide.pdf 9

[31] S. Hao, D. Li, W. G. J. Halfond, R. Govindan, Estimating mobile application energy consumption using program analysis, in: 2013 35th International Conference on Software Engineering (ICSE), 2013, pp. 92–101. doi:10.1109/ICSE.2013.6606555. 10, 39

[32] R. J. Behrouz, A. Sadeghi, J. Garcia, S. Malek, P. Ammann, Ecodroid: An approach for energy-based ranking of android apps, in: 2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software, 2015, pp. 8–14. doi:10.1109/GREENS.2015.9. 10

[33] S. A. Chowdhury, A. Hindle, Greenoracle: Estimating software energy consumption with energy measurement corpora, in: Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 49–60. doi:10.1145/2901739.2901763.
URL https://doi.org/10.1145/2901739.2901763 10

[34] Microsoft, Joulemeter.
URL https://research.microsoft.com/en-us/projects/joulemeter 10

[35] J. Reich, M. Goraczko, A. Kansal, Sleepless in seattle no longer, in: 2010 USENIX Annual Technical Conference (USENIX ATC 10), USENIX Association, 2010.
URL https://www.usenix.org/conference/usenix-atc-10/sleepless-seattle-no-longer 10

[36] A. Noureddine, R. Rouvoy, L. Seinturier, A review of energy measurement approaches, ACM SIGOPS Operating Systems Review 47 (3) (2013) 42–49. 10

[37] T. Do, S. Rawshdeh, W. Shi, ptop: A process-level power profiling tool, 2009. 11

[38] J. Pouwelse, K. Langendoen, H. Sips, Dynamic voltage scaling on a low-power microprocessor, in: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, MobiCom '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 251–259. doi:10.1145/381677.381701.
URL https://doi.org/10.1145/381677.381701 12

[39] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, A. Lucia, Software-based energy profiling of android apps: Simple, efficient and reliable?, 2017. doi:10.1109/SANER.2017.7884613. 13

[40] S. Schubert, D. Kostic, W. Zwaenepoel, K. G. Shin, Profiling software for energy consumption, in: 2012 IEEE International Conference on Green Computing and Communications, 2012, pp. 515–522. doi:10.1109/GreenCom.2012.86. 13

[41] K. J. J.Sharp, Thermal Imaging Techniques to Survey and Monitor Animals in the Wild, Academic Press, 2006. 13

[42] M. E. T. V. D. F. P. B. P. F. A. Duarte, L. Carrão, H. Almeida, Segmentation algorithms for thermal images, Procedia Technology 16 (2014) 1560–1569. 14

[43] P. W. Kruse, et al., Uncooled thermal imaging: arrays, systems, and applications, Vol. 2003, SPIE press Bellingham, WA, 2001. 14

[44] V. Pašagić, M. Mužević, D. Kelenc, Infrared thermography in marine applications, Brodogradnja (brodogradnja@hrbi.hr); Vol.59 No.2 59. 14

[45] Y. Abdelrahman, E. Velloso, T. Dingler, A. Schmidt, F. Vetere, Cognitive heat: exploring the usage of thermal imaging to unobtrusively estimate cognitive load, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1 (3) (2017) 33. 14

[46] C. Hildebrandt, C. Raschner, K. Ammer, An overview of recent application of medical infrared thermography in sports medicine in austria, Sensors (Basel, Switzerland) 10 (2010) 4700–15. doi:10.3390/s100504700. 14

[47] R. Gade, T. B. Moeslund, Thermal cameras and applications: a survey, Machine vision and applications 25 (1) (2014) 245–262. 15

[48] W. Chiu, P. Lin, H. Chiou, W. Lee, C. Lee, Y. Yang, H. Lee, M. Hsieh, C. Hu, Y. Ho, W. Deng, C. Hsu, Infrared thermography to mass-screen suspected sars patients with fever, Asia Pacific Journal of Public Health 17 (1) (2005) 26–28. 16

[49] W. K. Wong, P. N. Tan, C. K. Loo, W. S. Lim, An effective surveillance system using thermal camera (2009) 13–17. 16

[50] R. Ishimwe, K. Abutaleb, F. Ahmed, Applications of thermal imaging in agriculture - a review, Advances in Remote Sensing 3 (2014) 128–140. doi:10.4236/ars.2014.33011. 16

[51] A. Pathak, Y. C. Hu, M. Zhang, Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices, in: Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, Association for Computing Machinery, New York, NY, USA, 2011. doi:10.1145/2070562.2070567.
URL https://doi.org/10.1145/2070562.2070567 16

[52] S. Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Henkel, S. X.-D. Tan, Post-silicon heat-source identification and machine-learning-based thermal modeling using infrared thermal imaging, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 40 (4) (2021) 694–707. doi:10.1109/TCAD.2020.3007541. 16

[53] F. Beneventi, A. Bartolini, P. Vivet, L. Benini, Thermal analysis and interpolation techniques for a logic + wideio stacked dram test chip, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35 (2015) 1–1. doi:10.1109/TCAD.2015.2474382. 17

[54] R. Cochran, S. Reda, Spectral techniques for high-resolution thermal characterization with limited sensor data, in: 2009 46th ACM/IEEE Design Automation Conference, 2009, pp. 478–483. doi:10.1145/1629911.1630037. 17

# BIBLIOGRAPHY


[55] J. Ranieri, A. Vincenzi, A. Chebira, D. Atienza, M. Vetterli, Eigenmaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors, in: DAC Design Automation Conference 2012, 2012, pp. 636–641. doi:10.1145/2228360.2228475. 17

[56] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, M. Pedram, Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor, 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (2013) 242–247. 17

[57] Q. Xie, M. J. Dousti, M. Pedram, Therminator: a thermal simulator for smartphones producing accurate chip and skin temperature maps. 17

[58] F. Paterna, J. Zanotelli, T. S. Rosing, Ambient variation-tolerant and inter components aware thermal management for mobile system on chips. 17

[59] D. J. Malan, M. D. Smith, Host-based detection of worms through peer-to-peer cooperation, in: Proceedings of the 2005 ACM Workshop on Rapid Malcode, WORM '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 72–80. doi:10.1145/1103626.1103641.
URL https://doi.org/10.1145/1103626.1103641 17

[60] R. Mittal, A. Kansal, R. Chandra, Empowering developers to estimate app energy consumption, in: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Mobicom '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 317–328. doi:10.1145/2348543.2348583.
URL https://doi.org/10.1145/2348543.2348583 17

[61] C. Herglotz, S. Grosche, A. Bharadwaj, A. Kaup, Component-wise power estimation of electrical devices using thermal imaging, IEEE Transactions on Consumer Electronics 67 (4) (2021) 383–392. doi:10.1109/TCE.2021.3122076. 17

[62] G. Satchell, M. McGrath, J. Dixon, T. Pfau, R. Weller, Effects of time of day, ambient temperature and relative humidity on the repeatability of infrared thermographic imaging in horses, Equine Veterinary Journal 47 (S48) (2015) 13–14. arXiv:https://beva.onlinelibrary.wiley.com/doi/pdf/10.1111/evj.12486_30, doi:https://doi.org/10.1111/evj.12486_30.
URL https://beva.onlinelibrary.wiley.com/doi/abs/10.1111/evj.12486_30 18

[63] A. Szentkuti, H. Kavanagh, S. Grazio, Infrared thermography and image analysis for biomedical use, Periodicum Biologorum 113 (2011) 385–392. 18

[64] D. T. T. Limited, Smartphone batteries: better but no breakthrough, https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Technology-Media-Telecommunications/gx-tmt-pred15-smartphone-batteries.pdf (2015). 19

[65] C. M. Kitchen, Nonparametric vs parametric tests of location in biomedical research, American Journal of Ophthalmology 147 (4) (2009) 571–572. doi:10.1016/j.ajo.2008.06.031.
URL https://doi.org/10.1016/j.ajo.2008.06.031 23

[66] M. Wellons, Stefan–boltzmann law, Introduction to Quantum Mechanics 1. 25

[67] I. Lourek, M. Tribeche, Thermodynamic properties of the blackbody radiation: A kaniadakis approach, Physics Letters A 381 (5) (2017) 452–456. doi:https://doi.org/10.1016/j.physleta.2016.12.019.
URL https://www.sciencedirect.com/science/article/pii/S0375960116320060 26

[68] Y.-H. Ma, D. Xu, H. Dong, C.-P. Sun, Optimal operating protocol to achieve efficiency at maximum power of heat engines, Physical Review E 98 (2) (2018) 022133. 32

[69] F. S. GUTHERY, R. L. BINGHAM, A primer on interpreting regression models, The Journal of Wildlife Management 71 (3) (2007) 684–692. arXiv:https://wildlife.onlinelibrary.wiley.com/doi/pdf/10.2193/2006-285, doi:https://doi.org/10.2193/2006-285.
URL https://wildlife.onlinelibrary.wiley.com/doi/abs/10.2193/2006-285 34

[70] C. Wilke, S. Richly, S. Götz, C. Piechnick, U. Assmann, Energy consumption and efficiency in mobile applications: A user feedback study, 2013. doi:10.1109/GreenCom-iThings-CPSCom.2013.45. 38

[71] F. Palomba, D. Di Nucci, A. Panichella, A. Zaidman, A. Lucia, On the impact of code smells on the energy consumption of mobile applications, Information and Software Technology 105. doi:10.1016/j.infsof.2018.08.004. 39

[72] H. Flores, J. Hamberg, X. Li, T. Malmivirta, A. Zuniga, E. Lagerspetz, P. Nurmi, Estimating energy footprint using thermal imaging, GetMobile: Mobile Computing and Communications 23 (3) (2020) 5–8. 39

[73] J. McChesney, N. Wang, A. Tanwer, E. De Lara, B. Varghese, Defog: fog computing benchmarks, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 47–58. 41

[74] S. Das, S. Chapman, J. Christopher, M. R. Choudhury, N. W. Menzies, A. Apan, Y. P. Dang, Uav-thermal imaging: A technological breakthrough for monitoring and quantifying crop abiotic stress to help sustain productivity on sodic soils – a case review on wheat, Remote Sensing Applications: Society and Environment 23 (2021) 100583. doi:https://doi.org/10.1016/j.rsase.2021.100583.
URL https://www.sciencedirect.com/science/article/pii/S2352938521001191 58

[75] C. Yuan, Z. Liu, Y. Zhang, Fire detection using infrared images for uav-based forest fire surveillance, in: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 2017, pp. 567–572. doi:10.1109/ICUAS.2017.7991306. 58

[76] D. A. Rahman, A. A. A. F. Rahman, Performance of unmanned aerial vehicle with thermal imaging, camera trap, and transect survey for monitoring of wildlife, IOP Conference Series: Earth and Environmental Science 771 (1) (2021) 012011. doi:10.1088/1755-1315/771/1/012011.
URL https://doi.org/10.1088/1755-1315/771/1/012011 58

[77] L. Santesteban, S. Di Gennaro, A. Herrero-Langreo, C. Miranda, J. Royo, A. Matese, High-resolution uav-based thermal imaging to estimate the instantaneous and seasonal variability of plant water status within a vineyard, Agricultural Water Management 183 (2017) 49–59, special Issue: Advances on ICTs for Water Management in Agriculture. doi:https://doi.org/10.1016/j.agwat.2016.08.026.
URL https://www.sciencedirect.com/science/article/pii/S0378377416303201 58

[78] U. Sreedhar, C. Krishnamurthy, K. Balasubramaniam, V. Raghupathy, S. Ravisankar, Automatic defect identification using thermal image analysis for online weld quality monitoring, Journal of Materials Processing Technology 212 (7) (2012) 1557–1566. doi:https://doi.org/10.1016/j.jmatprotec.2012.03.002.
URL https://www.sciencedirect.com/science/article/pii/S092401361200074X 58

[79] N. Khera, S. Khan, O. Rahman, Valve regulated lead acid battery diagnostic system based on infrared thermal imaging and fuzzy algorithm, International Journal of System Assurance Engineering and Management 11. doi:10.1007/s13198-020-00958-z. 59

[80] R. Furstenberg, C. Kendziora, M. Papantonakis, S. V. Stepnowski, J. Stepnowski, V. Nguyen, M. Rake, R. A. McGill, Stand-off detection of trace explosives by infrared photo-thermal spectroscopy, in: 2009 IEEE Conference on Technologies for Homeland Security, 2009, pp. 465–471. doi:10.1109/THS.2009.5168074. 59

[81] J. Seo, I.-J. Chung, Face liveness detection using thermal face-cnn with external knowledge, Symmetry 11 (3). doi:10.3390/sym11030360.
URL https://www.mdpi.com/2073-8994/11/3/360 59

# 9

# Appendix

## 9.1 Licence

**Non-exclusive licence to reproduce thesis and make thesis public**

**I, Hashim Hashimov,**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, **Modelling Energy Consumption using Thermal Imaging**, supervised by Huber Flores.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Hashim Hashimov
**08/08/2022**