CHAPTER 1

Prototype Design

As discussed in the Literature Review, using an Infrared Array Sensor (IAR) appear to be the most viable way to achieve the high-level goals of this project. Thermosense [4], the primary occupancy sensor in the IAR space, used the low-cost Panasonic Grid-EYE sensor for this task. This sensor, costing around $30USD, appears to be a prime candidate for use in this project, as it satisfied low-cost criteria, as well as being proven by Thermosense to be effective in this space. However, while still available for sale in the United States, we were unable to order the sensor for shipping to Australia due to export restrictions outside of our control. While such restrictions would be circumventable with sufficient effort, using a sensor with such restrictions in place goes against an implicit criteria of the parts used in the project being relatively easy to acquire.

This forced us to search for alternative sensors in the space that fulfill similar criteria but were more broadly available. The sensor we settled on was the Melexis MLX90620 (*Melexis*) [5], an IAR with similar overall qualities that differed in several important ways; it provides a 16×4 grid of thermal information, it has an overall narrower field of view and it sells for approximately $80USD. Like the Grid-EYE , the *Melexis* sensor communicates over the 2-wire I^2C bus, a low-level bi-directional communication bus widely used and supported in embedded systems.

In an idealized version of this occupancy system, much like Thermosense this system would include wireless networking and a very small form factor. However, due to time and resource constraints, the scope of this project has been limited to a minimum viable implementation. Appendix Chapter **??** on page ?? discusses in detail how the introduction of new open standards in the Wireless Personal Area Network space could be used in future systems to provide robust, decentralized networking of future occupancy sensors. This prototype architecture has been designed such that a clear path to the idea system architecture discussed therein is available.

Analysis Tier	Raspberry Pi B+
Preprocessing Tier	Arduino Uno R3
Sensing Tier	Melexis MLX90620 & PIR

Table 1.1: Hardware tiers

1.1 Hardware

As reliability and future extensibility are core concerns of the project, a three-tiered system is employed with regards to the hardware involved in the system (Table 1.1). At the bottom, the Sensing Tier, we have the raw sensor, the Melexis MLX90620 (*Melexis*), which communicate over I^2C . Connected to these devices via those respective protocols is the Preprocessing Tier, run an embedded system. The embedded device polls the data from these sensors, performs necessary calculations to turn raw information into suitable data, and communicates this via Serial over USB to the third tier. The third tier, the Analysis Tier, is run on a fully fledged computer. In our prototype, it captures and stores both video data, and the Temperature and Motion data it receives over Serial over USB.

While at a glance this system may seem overly complicated, it ensures that a sensible upgrade path to a more feature-rich sensing system is available. In the current prototype, the Analysis Tier merely stores captured data for offline analysis, in future prototypes this analysis can be done live and served to interested parties over a RESTful API. In the current prototype, the Analysis and Sensing Tiers are connected by Serial over USB, in future prototypes, this can be replaced by a wireless mesh network, with many Preprocessing/Sensing Tier nodes communicating with one Analysis Tier node.

Due to low cost and ease of use, the Arduino platform was selected as the host for the Preprocessing Tier, and thus the low-level I^2C interface for communication to the *Melexis*. Initially, this presented some challenges, as the *Melexis* recommends a power and communication voltage of 2.6V, while the Arduino is only able to output 3.3V and 5V as power, and 5V as communication. Due to this, it was not possible to directly connect the Arduino to the *Melexis*, and similarly due to the two-way nature of the I^2C 2-wire communication protocol, it was also not possible to simply lower the Arduino voltage using simple electrical techniques, as such techniques would interfere with two-way communication.

A solution was found in the form of a I^2C level-shifter, the Adafruit "4-channel I2C-safe Bi-directional Logic Level Converter" [1], which provided a cheap method to bi-directionally communicate between the two devices at their own preferred voltages. The layout of the circuit necessary to link the Arduino

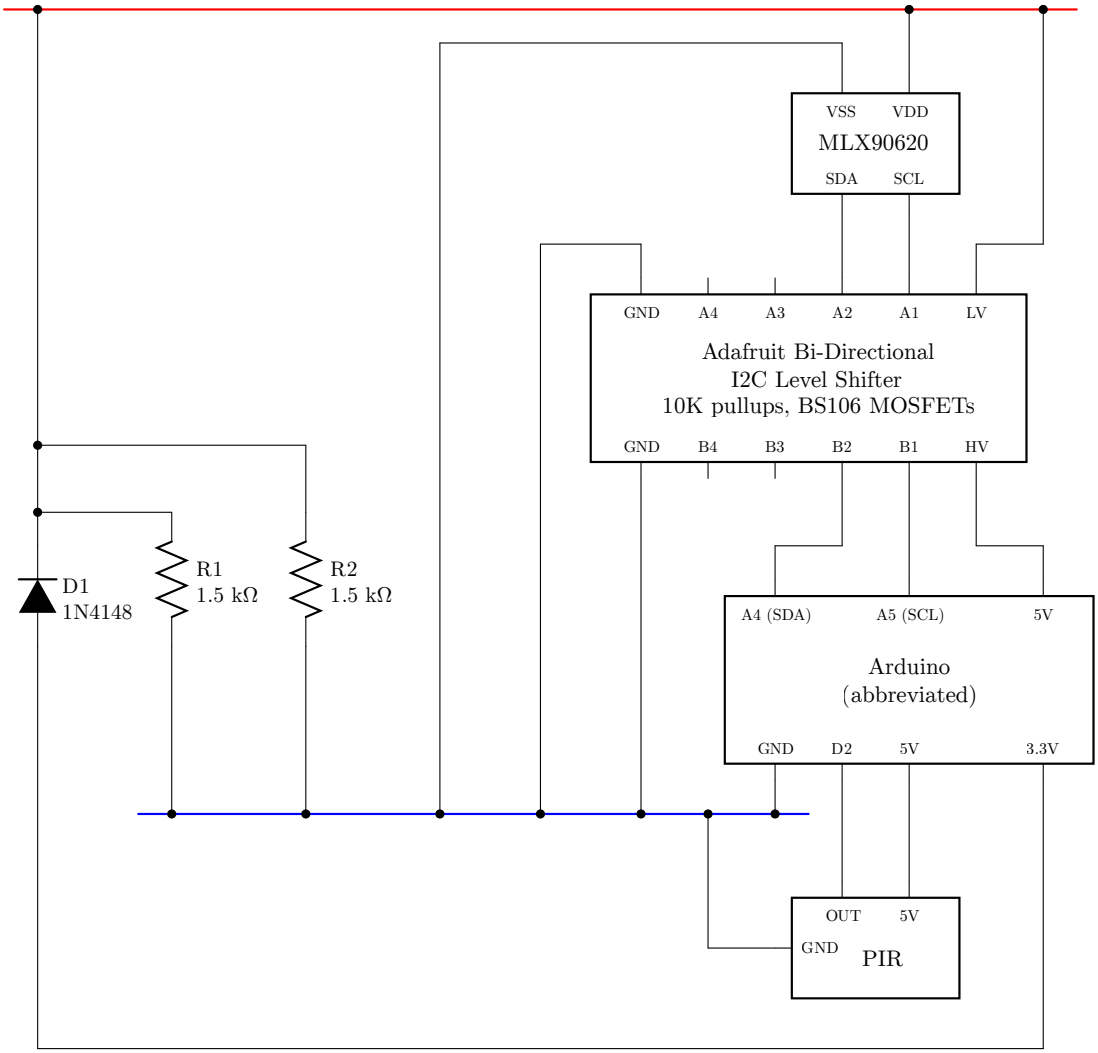

Figure 1.1: MLX90620, PIR and Arduino integration circuit

and the *Melexis* using this converter can be seen in Figure 1.1 on the preceding page.

Additionally, as used in the Thermosense paper, a Passive Infrared Sensor (PIR) motion sensor [2] was also connected to the Arduino . This sensor, operating at 5V natively, did not require any complex circuitry to interface with the Arduino . It is connected to digital pin 2 on the Arduino , where it provides a rising signal in the event that motion is detected, which can be configured to cause an interrupt on the Arduino . In the configuration used in this project, the sensor's sensitivity was set to the highest value and the timeout for re-triggering was set to the lowest value (approximately 2.5 seconds). Additionally, the continuous re-triggering feature (whereby the sensor produces continuous rising and falling signals for the duration of motion) was disabled using the provided jumpers.

For the Analysis Tier, the Raspberry Pi B+ was chosen, as it is a powerful computer capable of running Linux available for an extraordinarily low price. The Arduino is connected to the Raspberry Pi over USB, which provides it both power and the capacity to transfer data. In turn, the Raspberry Pi is connected to a simple micro-USB rechargeable battery pack, which provides it with power, and subsequently the Arduino and sensors.

1.2 Software

To calculate the final temperature values that the Melexis MLX90620 (*Melexis*) offers, a complex initialisation and computational process must be followed, which is specified in the sensor's datasheet [5]. This process involves initialising the sensor with values attained from a separate on-board I^2C EEPROM, then retrieving a variety of normalisation and adjustment values, along with the raw sensor data, to compute the final temperature result.

The basic algorithm to perform this normalisation was based upon code by users "maxbot", "IIBaboomba", "nseidle" and others on the Arduino Forums [3] and was modified to operate with the newer Arduino "Wire" I^2C libraries released since the authors' posts. In pursuit of the project's aims to create a more approachable thermal sensor, the code was also restructured and rewritten to be both more readable, and to introduce a set of features to make the management of the sensor data easier for the user, and for the information to be more human readable.

The first of the features introduced was the human-readable format for serial transmission. This allows the user to both easily write code that can parse the serial to acquire the serial data, as well as examine the serial data directly with

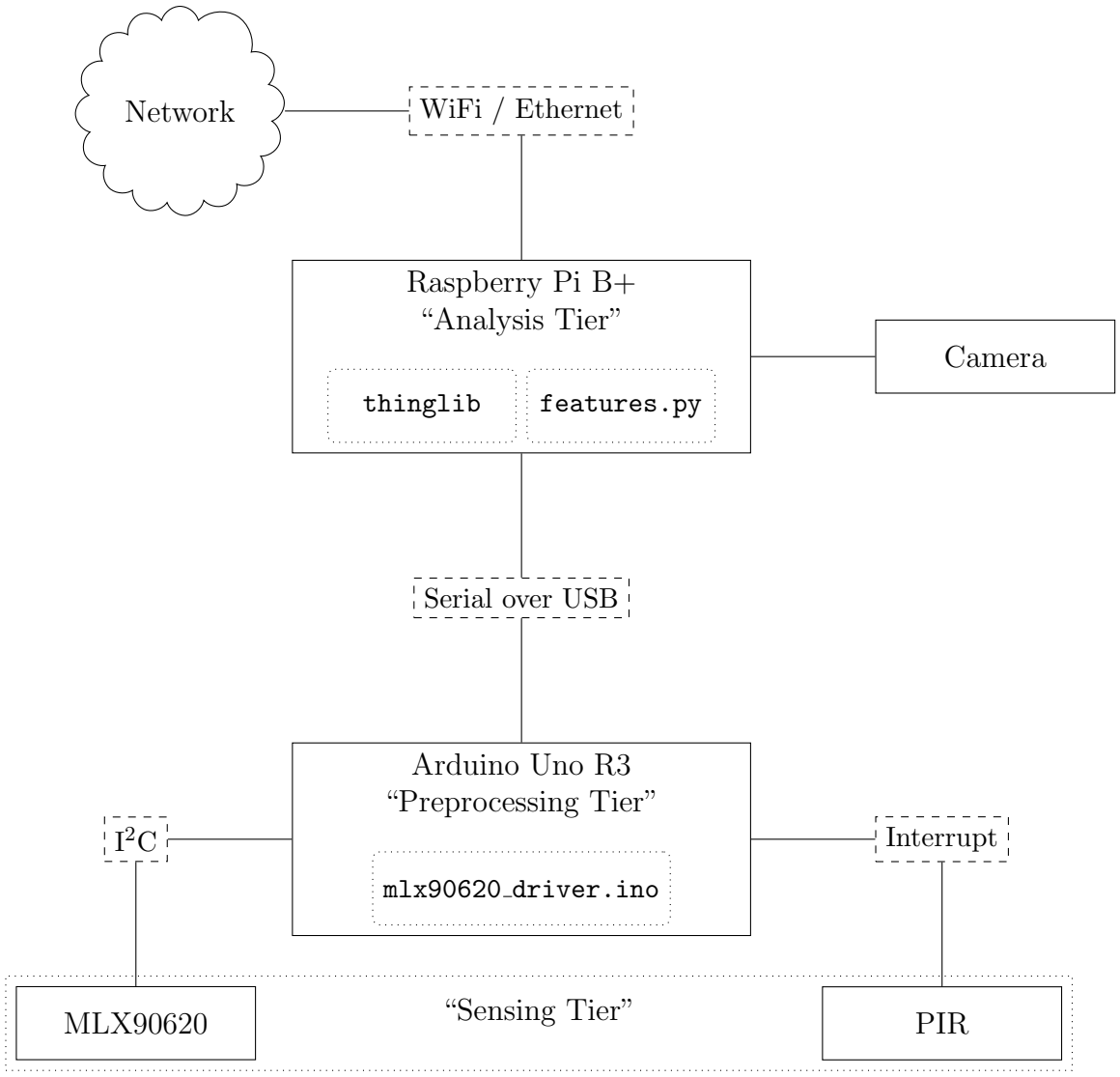

Figure 1.2: Prototype system architecture

```
INIT 0
INFO START
DRIVER MLX90620
BUILD Feb  1 2015 00:00:00
IRHZ 1
INFO STOP
ACTIVE 33

START 34
MOVEMENT 0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
STOP 97
```

Figure 1.3: Initialisation sequence and thermal packet

ease. When the Arduino first boots running the software, the output in Figure 1.3 is output. This specifies several things that are useful to the user; the attached sensor ("DRIVER"), the build of the software ("BUILD") and the refresh rate of the sensor ("IRHZ"). Several different headers, such as "ACTIVE" and "INIT" specify the current millisecond time of the processor, thus indicating how long the execution of the initialisation process took (33 milliseconds).

Once booted, the user is able to send several one-character commands to the sensor to configure operation, which are described in Table **??** on page ??. Depending on the sensor configuration, IR data may be periodically output automatically, or otherwise manually triggered. This IR data is produced in the packet format described in Figure 1.3. This is a simple, human readable format that includes the millisecond time of the processor at the start and end of the calculation, if the Passive Infrared Sensor (PIR) has seen any motion for the duration of the calculation, and the 16x4 grid of calculated temperature values.

1.3 Sensor Properties

In order to best utilize the Melexis MLX90620 (*Melexis*), we must first understand the properties it exhibits, and their potential affects on our ability to perform person related measurements. These properties can be broadly separated into three different categories; bias, noise and sensitivity. A broad range of data was collected with the sensor in a horizontal orientation using various sources of heat and cold to determine these properties. This experimental setup is described in Figure 1.4 on the following page.

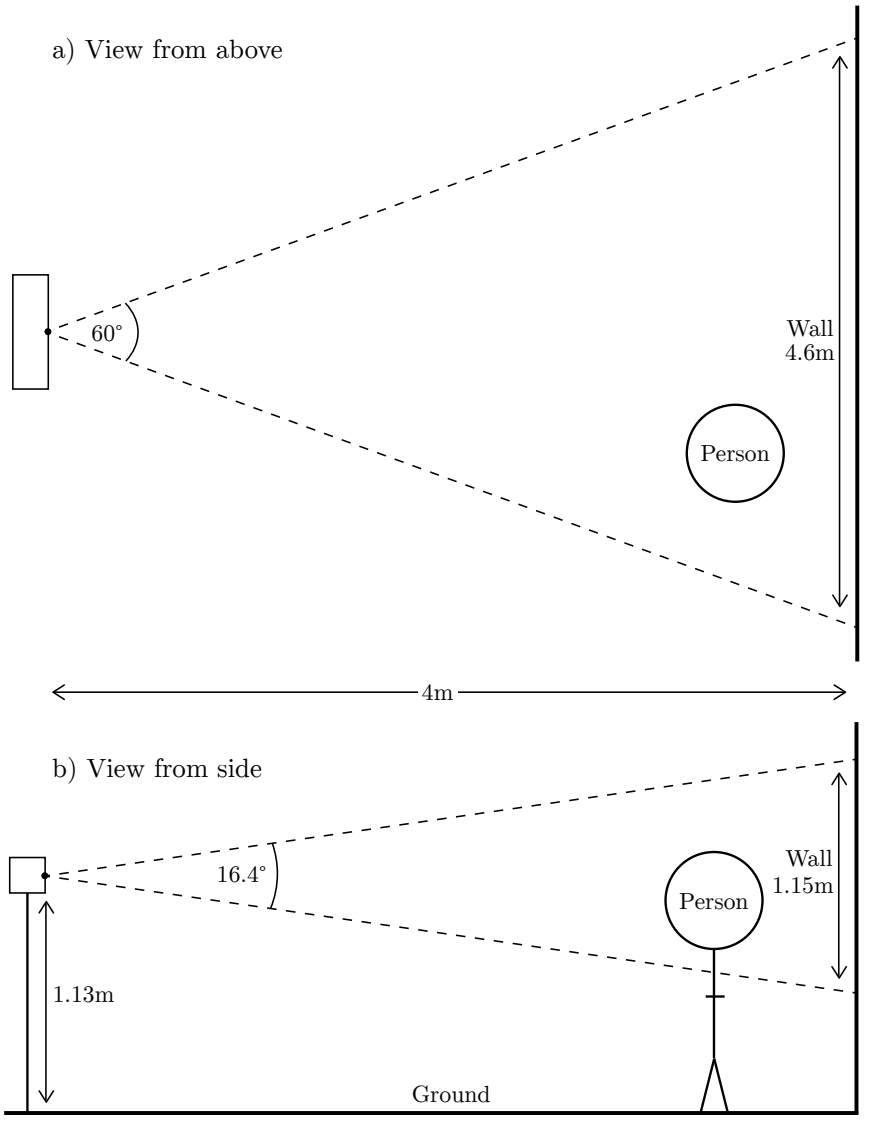

Figure 1.4: Experiment setup to determine sensor properties

1.3.1 Bias

When receiving no infrared radiation, the sensor should indicate a near-zero temperature. If in such conditions it does not, that indicates that the sensor has some level of bias in its measurement values. We attempted to investigate this bias by performing thermal captures of the night sky. While this does not completely remove the infrared radiation, it does remove a significant proportion of it.

In Table 1.2 on the next page the thermal sensor was exposed to the night sky at a capture rate of 1Hz for 4 minutes, with the sensing results combined to create a set of means and standard deviations to indicate the pixels at "rest". The average temperature detected was 11.78°C, with the standard deviation remaining less than 0.51°C over the entire exposure period. The resultant thermal map shows that pixels centered around the four "primary" pixels in the centre maintain a similar temperature around 9°C, with temperatures beginning to deviate as they became further from the center.

14.95 0.51	14.33 0.27	12.34 0.27	8.77 0.33	8.15 0.31	10.84 0.38	9.02 0.26	7.79 0.37	6.67 0.27	9.63 0.29	9.29 0.26	8.24 0.27	9.84 0.25	14.28 0.33	14.92 0.3	13.16 0.25
14.54 0.34	15.62 0.31	12.73 0.23	11.51 0.27	11.79 0.26	11.47 0.27	11.43 0.29	9.02 0.35	8.57 0.23	11.15 0.23	10.64 0.22	10.3 0.24	12.09 0.22	14.49 0.26	14.88 0.31	14.71 0.36
18.25 0.45	16.62 0.31	14.15 0.24	11.97 0.34	13.11 0.3	12.64 0.22	10.66 0.23	9.15 0.24	9.58 0.28	11.95 0.28	11.22 0.24	11.52 0.36	11.11 0.23	12.59 0.25	14.44 0.31	13.35 0.28
16.02 0.28	16.81 0.36	15.0 0.25	11.53 0.28	10.18 0.29	12.2 0.25	11.78 0.29	8.36 0.31	8.15 0.33	10.36 0.32	10.74 0.31	8.25 0.36	9.99 0.35	12.42 0.38	11.39 0.4	11.06 0.34

Table 1.2: Mean and standard deviations for each pixel at rest

1.3.2 Noise

1.3.3 Sensitivity

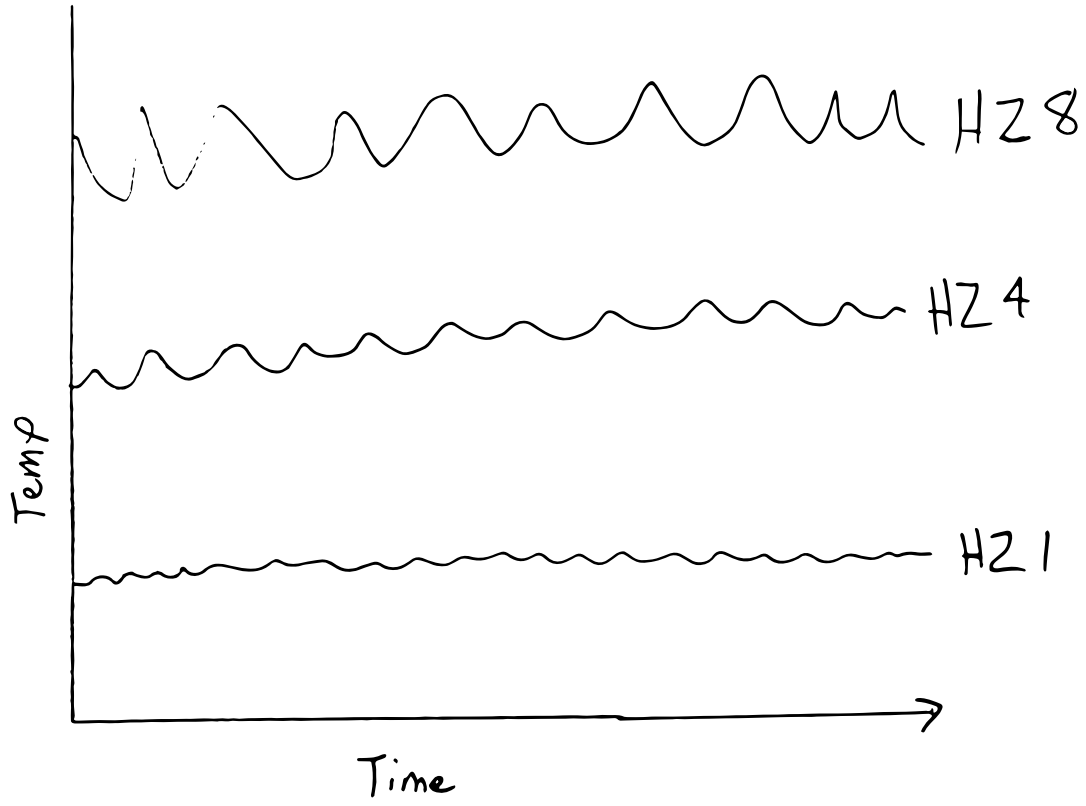

Figure 1.5: Comparison of noise levels at the *Melexis*' various sampling speeds

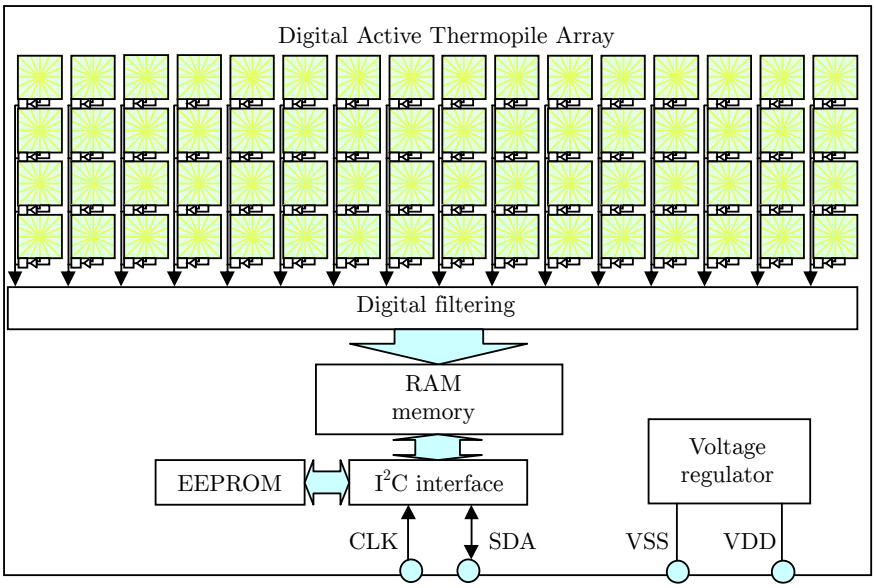

Figure 1.6: Block diagram for the *Melexis* taken from datasheet [5]

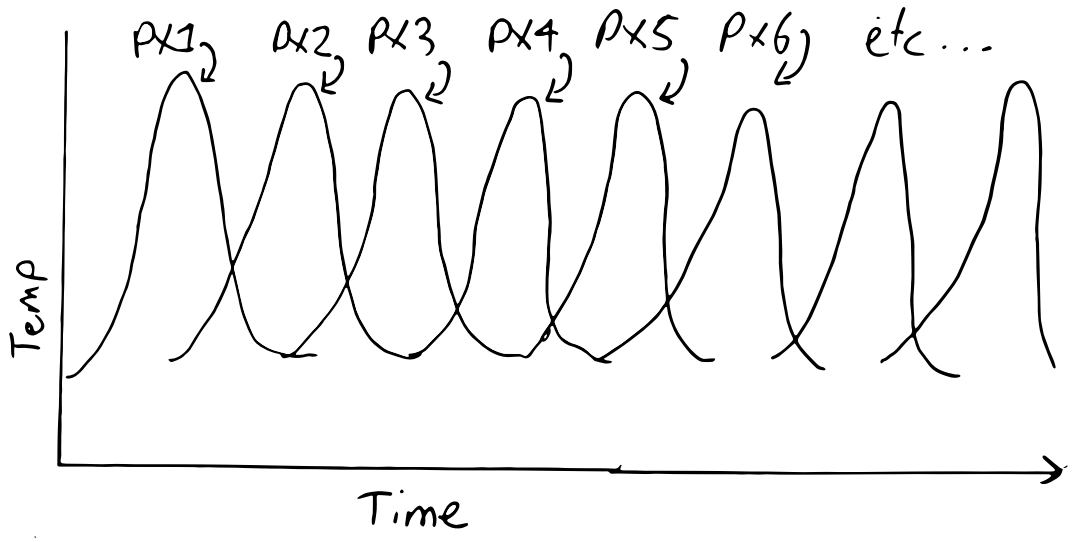

Figure 1.7: Different *Melexis* pixel temperature values as hot object moves across row

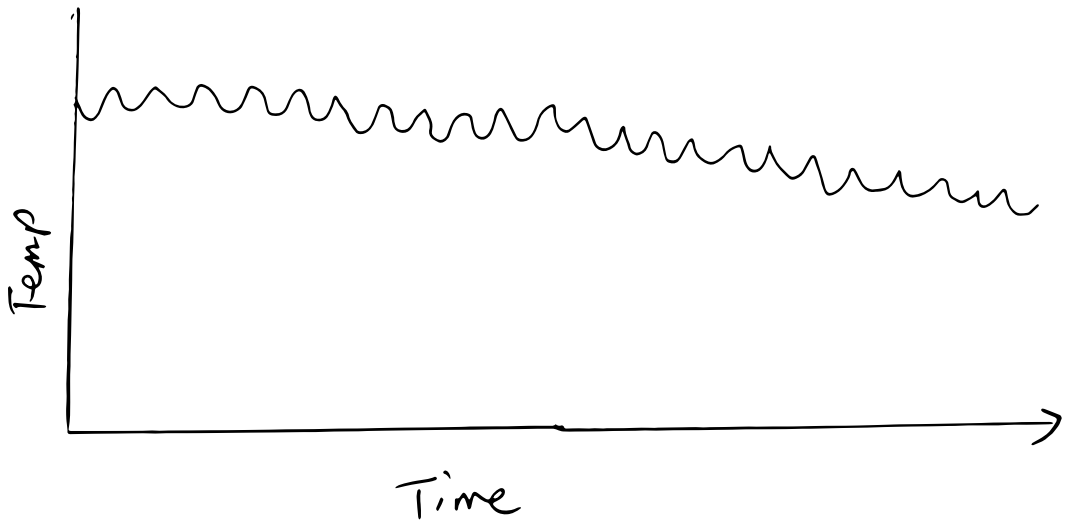

Figure 1.8: Variation in temperature detected for hot object at 1Hz sampling ration

Bibliography

[1] Adafruit. 4-channel I2C-safe bi-directional logic level converter - BSS138 (product ID 757). http://www.adafruit.com/product/757. Accessed: 2015-01-07.

[2] Adafruit. PIR (motion) sensor (product ID 189). http://www.adafruit.com/product/189. Accessed: 2015-02-08.

[3] Arduino Forums. Arduino and MLX90620 16X4 pixel IR thermal array. http://forum.arduino.cc/index.php/topic,126244.0.html, 2012. Accessed: 2015-01-07.

[4] Beltran, A., Erickson, V. L., and Cerpa, A. E. ThermoSense: Occupancy thermal based sensing for HVAC control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–8.

[5] Melexis. Datasheet IR thermometer 16X4 sensor array MLX90620. http://www.melexis.com/Asset/Datasheet-IR-thermometer-16X4-sensor-array-MLX90620-DownloadLink-6099.aspx, 2012. Accessed: 2015-01-07.