THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MODELING IOT APPLICATION INTERACTIONS IN PHYSICAL SPACES

ELIF ERDOGDU
FALL 2019

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Patrick Drew McDaniel
William L. Weiss Professor of Information and Communications Technology in the School of Computer Science and Engineering
Thesis Supervisor

John Joseph Hannan
Associate Department Head of the School of Computer Science and Engineering
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

IoT (Internet of Things) is emerging as a powerful sector of technology, providing the opportunity to collect a wealth of valuable information and automate mundane tasks. Now more than ever, IoT devices are playing significant roles in various settings such as monitoring industrial automation, tracking human health, and providing convenience and security in our homes. The number of deployed IoT devices is growing exponentially: as of December 2018, there exist 8 Billion IoT devices, and the number is estimated to double by 2020 [1]. As the number of connected devices increases exponentially fast, and the IoT attack surface continues to expand, securely connecting these systems becomes critically important. Security researchers have been studying vulnerabilities in IoT devices and systems, and a number of their findings suggest that their proliferation will lead to security threats introduced by the physical space these devices share. Research has indicated that simple IoT devices positioned near each other, paired with IoT applications, can interact in unintended and harmful ways via their physical environment [2] [3].The created interactions are classified as negative if two or more devices' correct operation result in an unintended behavior, provoking harm for user's security, safety and privacy. Although researchers have demonstrated the possibility of these interactions through code analysis, whether they can occur in the physical world is still an open question. This work examines the negative interactions articulated in research space and investigates their feasibility in the real world through the physical modeling of IoT devices. We provide an approach to model IoT devices with commodity Arduino microcontrollers and sensors. We build five IoT devices and design a suite of scenarios that demonstrate negative device interactions in realistic settings. The results of this work confirm that IoT devices interact with each other in physical spaces and cause security, safety and privacy harm to the user.

The main contributions of this work are: (1) We show that the theorized negative physical interactions between IoT devices do occur, especially with high-precision sensors, (2) we introduce a process to accurately model IoT devices with commercial sensors and Arduino microcontrollers, and (3) we augment the IoTBench suite of IoT applications by incorporating test cases used in this work.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank Dr. Patrick McDaniel for introducing me to the world of research and also for entrusting me with this project.

I would also like to recognize Dr. Berkay Celik's role in encouraging me to explore the emerging field of Internet of Things. Additionally, my gratitude goes to all members of the SIIS Laboratory for their assistance and support in the completion of my thesis.

Remaining in the academic realm, I also owe thanks to my Honors Advisor John Hannan who helped me throughout my undergraduate career.

Finally, thank you to my family for your unconditional love. I could not have completed this without your continued support.

Thesis Statement

IoT devices can have negative interactions through the medium of physical spaces in ways that threaten consumer security, safety and privacy.

Chapter 1

Introduction

Over the past few years, the popularity of device automation has exploded. Gartner, a leading research company, predicts that 20.4 billion IoT devices will be in use worldwide by 2020 [1]. This exponential growth of IoT has brought new security challenges, leaving researchers with little time to determine and limit their impact. Until recently, devices around our homes operated in an isolated way, with little to no risk of interacting with other devices. Today, IoT connects these devices to each other and the internet, providing incredible convenience for end users. However, this convenience comes at the cost of introducing unintended and harmful interactions enabled through their physical environments.

IoT devices present a set of security threats distinct from those found on more traditional internet-connected devices such as laptops and desktop computers. Because they interact with physical entities like appliances inside of a home, existing computer security practices do not account for some of the scenarios presented by an IoT environment. Hence, a direct application of existing security software will not address these issues. Despite the significant amount of research focusing on IoT device security, there still exist recently discovered risks waiting to be further explored, one of them being the main focus of this paper, negative interactions in physical spaces [2].

The concept of "negative interactions between IoT devices" refers to unintended behavior caused by one or more devices, operating as programmed, affecting the physical environment in a way that causes another device monitoring that environment to trigger an action which could threaten the security, safety or privacy of the user. While the devices individually function just as instructed, the underlying cause of these interactions include a violation of assumptions of the device's environment. IoT devices, more specifically the data regarding the device's environment, are triggered by changes in the state of the physical spaces they are placed in. While this is a correct implementation, the physical environment can be influenced by other IoT devices to cause negative interactions not taken into account during the manufacturing of the device. The negative interactions between devices can have significant consequences such as unlocked doors and windows while using a smart kettle, gas leaks due to a smart fan trying to cool down the kitchen temperature, and displaying a bedroom in the middle of the night with automated lights and blinds.

Researchers have created methods to identify smart-device applications which could be subject to negative interactions through a physical space. One approach is a static analysis system, Soteria, which models interactions between devices through source code analysis [2]. Another solution is to control interactions through a framework, IoTMon, that assesses the safety of each inter-app interaction through Natural Language Processing [3].

This work examines the negative interactions articulated by these researchers, and investigates their feasibility in the real world. Our experiments confirm the interactions between IoT devices in physical spaces and verify the security, safety and privacy threats.

The contributions of this paper include the following: (1) we demonstrate the potential risks listed in previous work by researching a way to validate their findings with physical interactions, (2) we introduce a method to replicate an IoT device by implementing commmercial off-the-shelf sensors on Arduino microprocessors, and finally (3), we contribute to a worldwide IoT application repository, IoTBench, that serves as a collection of IoT applications that pose a challenge to the environment's security [4].

Chapter 2

Background

This chapter provides background information and summaries of important, relevant research studies regarding IoT devices and their security standpoint.

2.1 IoT devices

Internet of Things involves extending connectivity beyond the usual devices such as desktops, smartphones and tablets, to a traditionally disconnected everyday object (ie: refrigerators, thermostats, locks). These devices can be used in various locations: from industrial settings, to home environments. In this work we will focus more on consumer related IoT devices, meaning objects such as connected fridges, home alarms, smart locks, and windows. An IoT device is comprised of four main components: (1) sensors to collect data, (2) connectivity to the internet, (3) a back-end performing data processing, and (4) a user interface [5]. First, the sensors collect data from the environment where the device is located. Then, the data is sent to the cloud through the device's Internet connectivity feature. The cloud processes the data and makes decisions about whether rules need to be triggered, then send updated states to the devices. Once the data is received, the device performs the tasks assigned to it through the user interface. The end user can assign specific tasks through IoT applications and IFTTT rules (explained in the following section), and is also able to monitor and control the device remotely from the device's interface.

2.2 IoT Application Structures

One of the most important features of IoT is its adaptability - the ability for manufacturers, users, and third parties to combine the capabilities of devices through developing custom applications. IoT devices operate generally by connecting to IoT platforms called hubs, where the user can monitor and control device functionalities. Important IoT platforms include Amazon AWS IoT, Apple's HomeKit, Samsung SmartThings, and OpenHAB. These platforms most often use similar programming structures and can implement user specific automated tasks. Once a device is connected to the hub, the user installs or designs custom applications in the form of simple conditional statement chains. These conditional statements enable the user to trigger a service from the device to take the desired action, usually using the form of IFTTT (If This Then That). For example, an IFTTT condition may be: if the living room smart bulb senses motion after 7 pm, turn on the light, or if the door is left unlocked for longer than 10 minutes, lock the doors and close the windows. We use the idea of IFTTT in the following chapters to model real life IoT applications.

2.3 Related Work

Various work has been completed in the emerging field of smart home applications to ensure security. Recent research from the academic community has been focusing on the possible exploitation of these IoT apps with unauthorized access, causing information leaks [6].

One of the first in-depth security analysis of smart home specific applications discusses the significance of the vulnerabilities in smart home frameworks in 2016. Fernandes et al. found that of all the smart applications in the commercial store of SmartThings, more than half of the

applications are over privileged, granting full access to the connected device [7]. This finding raised attention to the security of the exponentially growing field of IoT.

More recently, in 2018 Celik et al. developed SainT, a tool that analyzes the functionality of an application and notifies the user in case any potential privacy risks are detected. The framework analyzes the source code of the IoT application and tracks each data flow throughout the execution. Any data labeled as sensitive is followed through the execution and reported if it transmits outside of the application [8].

Finally, Ding et al. capture a set of potential threats in Samsung SmartThings IoT platform through an IoT device control system IoTMon [3]. They study 185 SmartThings apps and find that 22% of the app interactions can result in environments that can be potentially exploited.

This thesis work constructs concrete sample scenarios of risky chains of events to establish whether application interactions can be modeled in real life settings and demonstrate the security challenges previously researched.

2.4 Examples of Negative interactions

In the previous section, we learned that IoT designs remain vulnerable and have important security challenges to overcome. Numerous use cases were identified where attackers can exploit the unsafe environments enabled by IoT devices.

Examples of such use cases mention multiple user-threatening settings that vary from the security of the network and malicious data analysis, to losing control of the device to attackers, to environmental security in the physical space. Here is a list of few examples to help the reader visualize the severeness of IoT device vulnerabilities.

- Example 1: Attackers can trigger a speech recognition system in popular devices such as Amazon's Alexa, Apple's Siri, Google Now. The speech of the attacker remains inaudible to humans through a DolphinAttack [9]. With inaudible voice commands, the consequences of this vulnerability can lead to creative attacks such as initiating calls on phones, turning devices to airplane mode, visiting malicious websites while dimming the screen brightness.

- Example 2: A simple connected light bulb can cause seizures to the user. The user downloads a malicious application from a third-party source to add to their IoT app control hub, with the intention of turning a light bulb on or off. If this app gives control to the the original developer, or the attacker, and if the attacker has previous knowledge of the user's condition, the user's health can be put in danger by generating strobe lights and forcing a seizure [4].

- Example 3: User downloads a third party app to monitor their door-lock battery. Regardless of how safe individual devices can be, the downloaded third party apps on IoT platforms can pose malicious intents that can be crucial to the safety of the user. Researchers were able to disguise a PIN recorder as a smart lock battery monitoring application. The attacker can monitor the PIN changes on the smart lock remotely because the SmartThings IoT platform grants access to random applications, without actually monitoring the intents [6].

2.5 Arduino Microcontroller

This work aims to model IoT devices by recreating their basic functionalities with Arduino microcontroller boards. An Arduino is an open source computing platform based on a microcontroller board. A basic model of Arduino can be used as a prototyping platform to sense and control the physical environment with a simple setup. The choice of the Arduino platform for this thesis is justified by its extensive online usage documentation, as well as the ease of access to and control of its hardware.

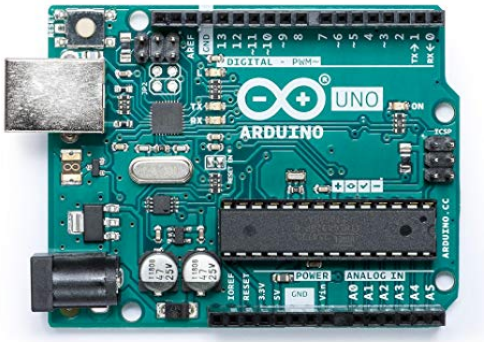

Figure 2.1: Arduino Uno Microcontroller Board

As of 2019, the average cost of one board is roughly $20, and the access to the Arduino development IDE requires no purchases.

2.5.1 Arduino Programming

Arduino is programmed through its software platform (IDE). The Arduino IDE is compatible to be used with any type of open source Arduino board and includes all the necessary libraries to run. The text editor used converts ".ino" code file written in a derivative of C/C++ , referred as a "sketch" in the Arduino environment, into C language. Sketches are then compiled by using avr-gcc and avr-g++, an open source compiler based on the GNU Compiler Collection (GCC) [7]. After the selection of the connected board, .c and .cpp files are sent onto the next compiler to become .hex files and uploaded to the board.

2.5.2 Sensors

Sensors are the foundation of IoT systems and are electronic devices that detect changes in an environment. These devices have practical features which can be used in many indoor and outdoor applications including security systems. In a smart home, common security sensors are used for smoke or gas detection and door/window opening control capabilities. These simple, low-cost devices can be programmed to report optical or electrical signals. Physical parameters such as temperature, humidity, light, and other environmental phenomena are measured and then converted to human-readable output. For the experiments described in this work, the following list of sensors and robotics are used to sculpt IoT devices [10].

- PIR Motion Sensor
  - Passive Infrared Sensors (PIR) are widely used in smart devices such as light bulbs, connected kitchen appliances, door locks etc. The PIR sensor detects motion through low-level radiation level changes emitted by warmth of an object. The radiation level changes indicate a moving body in the sensor's view range. The PIR sensor uses a digital pin to alert motion by outputting "HIGH" or "LOW" measurements. Finally, the sensor itself is quite cheap to find on the market, making it an ideal choice to model a low-cost motion detection for IoT devices.

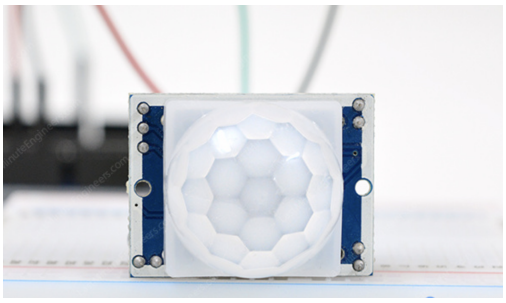

Figure 2.2: PIR Motion Sensor

- HC-SR04 Ultrasonic Sensor
  - This device can be used in security systems for detection of moving objects and employed with relation to smart door/window locks. The Ultrasonic Distance Sensor is used to detect range change in its 13 feet range, with an accuracy of 3mm. The two ultrasonic transducers convert signals to ultrasonic sound pulses and determine the distance pulses travel. Finally, the sensor itself is inexpensive, low power and can be used with batteries and Arduinos. It is important to keep in mind that the cheap manufacturing can affect the modeling of an IoT device.

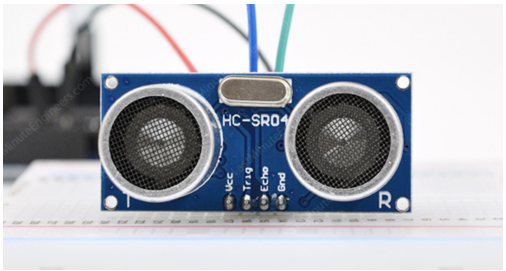

Figure 2.3: HC-SR04 Ultrasonic Sensor

- DHT11 Temperature and Humidity Sensor
  - The DHT11 Sensor is used to measure temperature and humidity in the environment. Smart devices such as thermostats or some smoke alarms can operate with this detector. The humidity sensing component perceives the change in humidity through the change in resistance between the two electrodes inside the sensor case. The temperature change is defined by a thermal resistor. The temperature measuring range is from $0°C$ to $50°C$ and 20% to 80%.

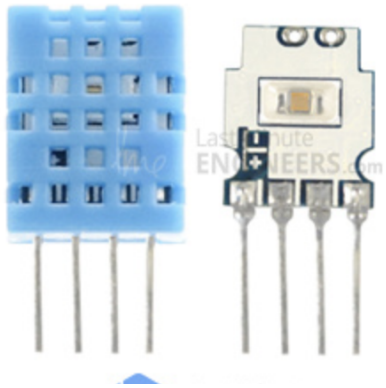

Figure 2.4: DHT11 Temperature and Humidity Sensor

- MQ2 Gas Sensor
  - The MQ2 Gas Sensor is typically utilized in smoke detecting security systems for detecting gas leakage such as LPG, i-butane, propane, methane, alcohol, hydrogen, smoke, and carbon monoxide. The sensor is enclosed in steel to prevent an explosion in case the heater element senses flammable gases. However, sensitivity of the sensor reduces significantly if it comes in contact with water, freezes, or is used in high gas concentration area for a long time.

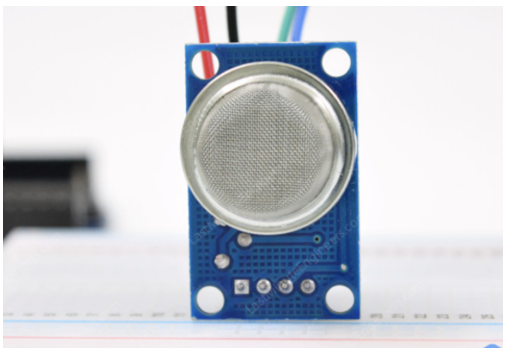

Figure 2.5: MQ2 Gas Sensor

- Power Relay Module
  - The power relay is an electronically operated switch that allows toggling a circuit. Smart devices that require any type of circuit switching, such as light bulbs, toasters, fans can be controlled remotely with a power relay. The module uses voltage and current that is much higher than the microcontroller can handle. This relay is ideal for small appliances but should be used with care and common sense if used with high voltages (more than 24V) [11]. The microcontroller outputs signals "HIGH" or "LOW" to the relay to close or open the circuit.

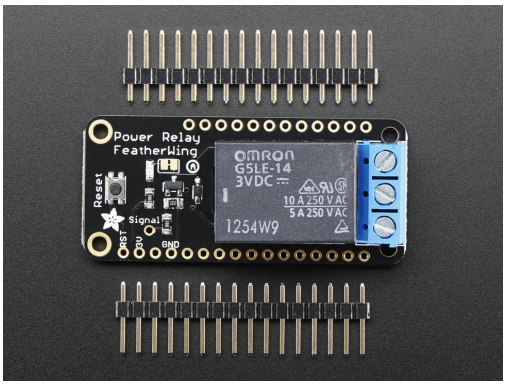

Figure 2.6: Power Relay FeatherWing

- Servo Motor
  - A servo motor can be used to add motion to a machine in a microcontroller project. It can be used to bring motion to devices, such as pushing a button, or flipping a pin connected to a device (i.e., kettle pin). Servo motors are useful in many robotics applications as one can control the positioning of the motor without having to build a motor controller. The motor consists of a small handle that can move from 0 degrees position to 90 degrees, to 180 degrees.

Figure 2.7: Servo Motor System

Chapter 3

Experimental Setup

The aim of this design is to provide a methodology to test negative interactions among IoT devices and to show that the potential threats reported by Ding et al. [3] in Chapter 2 can be in fact replicated in real life. In this paper we analyze IoT devices programmed with simple IFTTT structures and investigate their effects on other IoT devices. We demonstrate the reality and the extent of the negative interactions described in Table 3.1, by modeling applicable IoT environments.

Device 1 Action	Device 2 triggered sensor
Execute motion	Motion Sensor
Output humidity	Smoke Alarm
Output temperature	Smoke Alarm
Output temperature	Temperature Sensor

Table 3.1: List of Interactions

3.1 Modeling IoT devices

IoT devices' most important features include data collection through sensors, connectivity to the cloud, and the option to custom program the device itself. Hence, in this experiment we model connected appliances with Arduinos and low cost sensors. The implementation was preferred to be low cost to keep consistency with the practicality and simplicity of IoT devices. In each section, we describe the sensors used and the code for relative applications can be found in Appendix.

3.1.1 Smart Blinds

Smart blinds offer users the possibility for their blinds to be controlled through their smartphones. The user does not have to pull a cord to open or close the motorized blinds. Several design companies such Lutron and Somfy sell this product, and their systems are compatible with IoT hubs such as Google Home, Apple HomeKit etc. In order to model a smart blind technology, we used servo motors described in the previous chapter. The servo motor controls the positioning of a cardboard that acts as a pin holding all the slats together, imitating open blinds.

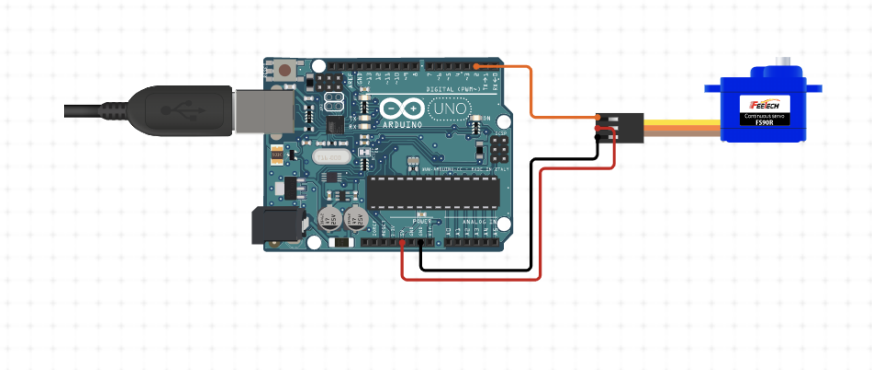

Figure 3.1: Blind Pin Schematic

Figure 3.2: Open Blinds Standing on Servo Motor

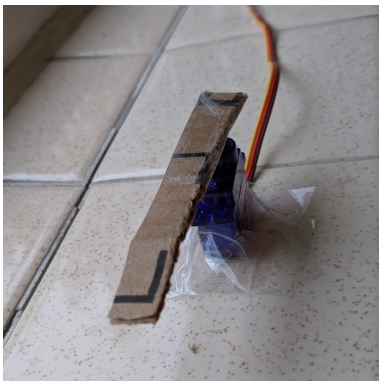

Figure 3.3: Servo Motor with Cardboard Pin

Figure 3.4: Closed blinds after Servo Motor Moves Pin

3.1.2 Smoke Alarm

Although temperature and humidity sensors are not required, many smoke alarms incorporate a heat sensing function [12]. Additionally, some smoke detectors can also be accidentally set off by high humidity. In order to model a smoke alarm, we use a MQ2 Smoke Sensor and a DHT11 temperature and humidity measurement sensor. To simulate the effects of an alarm going off we use a buzzer attachment on Arduino and as well as a servo motor to simulate doors and window unlocking.

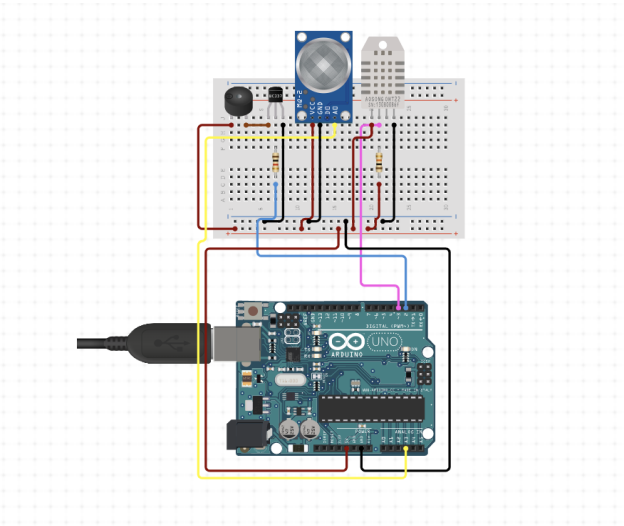

Figure 3.5: Smoke Alarm Schematic

3.1.3 Connected Kettle

In order to build a connected kettle the first implementation idea was to use a servo motor to push the kettle's switch up or down. Unfortunately the force of a turning servo motor remained

insufficient to push a kettle switch up or down. Due to the limited force of a servo motor, turning on a kettle switch with a servo motor was not feasible. Therefore we decided to use only a power relay that imitates the opening and closing of a circuit.

In addition to the relay, the Arduino board has a motion sensor attached to it to recreate a kettle turning on or off with a motion command. We use both PIR and ultrasonic motion sensors separately in the experiments.

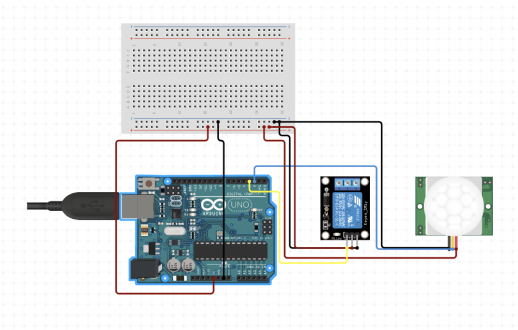

Figure 3.6: Kettle Schematic with a Power Relay and a PIR sensor

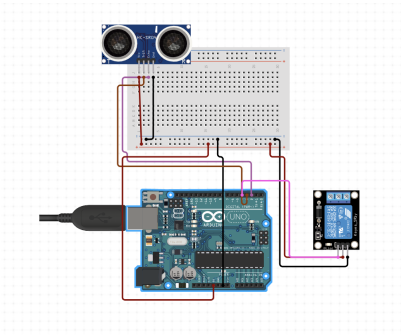

Figure 3.7: Kettle Schematic with a Power Relay and an Ultrasonic sensor

3.1.4 Automated Fan

The Automated Fan application is used in the second experiment in Section 4.2. The experiment depicts an application defined with the following IFTTT structure: if the temperature exceeds a certain threshold, turn on the fan and keep it on until temperature decreases. With the purpose of recreating this relation, we used a temperature sensor and a power relay on the same Arduino. The final Arduino product was able to model an automated fan and its application rules. However, to see the effects of an actual fan and its airflow, we used a mini portable desk fan.

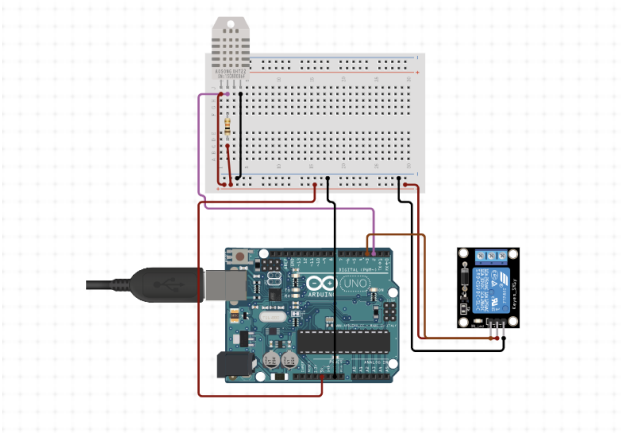

Figure 3.8: Automated Fan Schematic with a Power Relay. Note the sketch in Figure portrays DHT22 temperature sensor, instead of the actually used DHT11. This does not affect the setup.

3.1.5 Smart Bulb

In order to build a smart bulb, just as with the kettle, the first implementation idea was to connect a power relay to a lamp's cable and controlling the outlet power through the power relay. Unfortunately this did not work well in execution. The power relay should be used very carefully with voltages above 24 V, and the lamp attempted to use was working on 110 V.

Our second option was to implement a smart bulb with LED lights, powered only when motion is detected. For the purpose of enabling a motion activated light bulb effect, we used the following schematics.

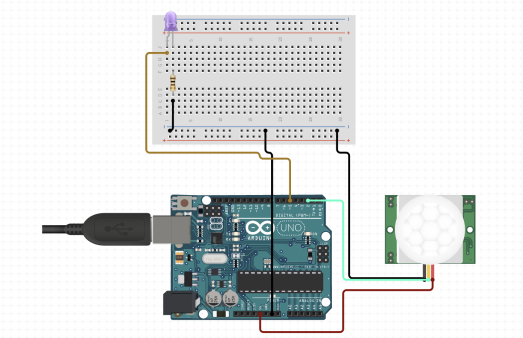

Figure 3.9: Smart Bulb Schematic with a PIR sensor

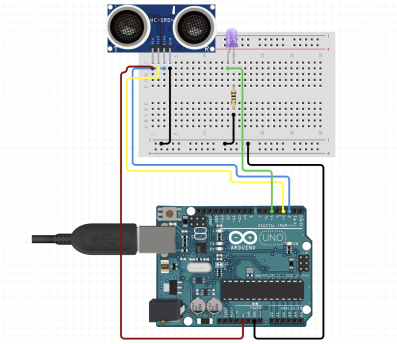

Figure 3.10: Smart Bulb Schematic with an Ultrasonic Sensor

3.1.6 Motion Sensing

Most of the devices described above make use of a motion detection system. Especially, the smart stove modeling we will use in section 4.2 will only consist of a motion sensor.

The commercially sold motion detection systems are designed by companies such as Samsung, Amazon, Apple and do not provide any details regarding what type of motion sensor these modules use. The provided features of the devices include the detection range, the set up instructions and no information about how the system works. Due to lack of information, we decided to represent device motion detection with two different sensors separately. First we use PIR motion and then we redo the same experiments with Ultrasonic Sensors.

Chapter 4

Experiments and Results

In the previous chapter, we introduced the equipment needed to model simple IoT devices. In this section, the reader will understand how the devices are put together. We also describe 3 sets of experiments to confirm if theorized research results remain true in our settings.

4.1 Experiment 1

4.1.1 Experiment Description

Negative interaction: Security

The first set of interactions tested happens between three separate IoT applications. The first application allows users to remotely open the blinds in the morning. This activates breakfast mode which we illustrate as a boolean value in the code. The second application is an automated kettle, which is programmed to turn on when it senses motion during breakfast mode. This scenario can be useful for the user as kettle starts boiling when the user walks into kitchen. Finally the third application is a smoke alarm detector which in case of an emergency, opens the doors and windows to enable easy escape.

Although the two events: opening the blinds and turning on a kettle, might seem like routine morning activities, the consequences of an interaction can be significant. The potential interaction implies that the kettle's motion sensor can pick up on the motion of the blinds if they are placed close enough. If the motion sensor gets activated while no one is around, it can lead an unsupervised working kettle with boiling hot water. Further theory from research papers show that the kettle humidity can also trigger a smoke alarm. In this experiment we set up the necessary devices as explained in previous chapters [3].

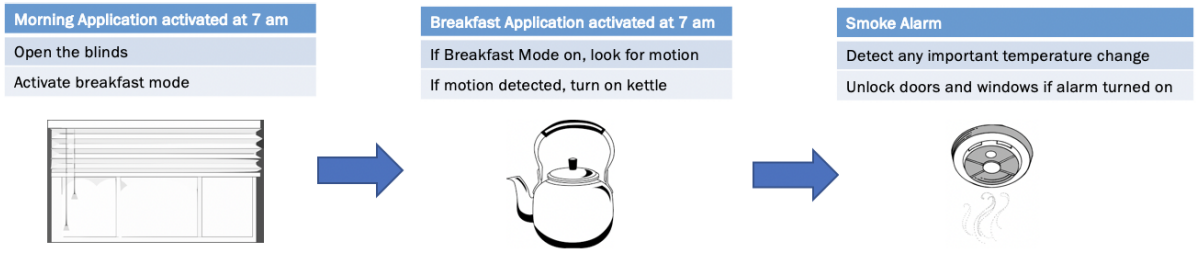

Figure 4.1: Experiment 1 Diagram

4.1.2 Observations and Results

Observations: First, we use a PIR motion sensor and the sensor does not detect any motion by the blinds. This is due to the fact that PIR motion sensors look for changes in infrared radiations. Consequently, we switch to an ultrasonic sensor to see if this time we can reproduce the expected outcome. The ultrasonic sensor does not register motion from blinds as it is a quick and steady move, insufficient to be registered as motion. The chain of risky events does not start with a PIR motion sensor.

We still want to check if a working kettle can trigger a smoke alarm. It is important to remember our modeled alarm has a humidity sensor and we want to see if it can be triggered with a kettle's boiling water outputs. In the measurements listed in Figure 4.2, we collected data over 3 states, before turning on the kettle, during the water boiling, and after turning it off. We notice that the humidity levels go from $40°F$ (regular room humidity), to 95% in less than 5 seconds. After

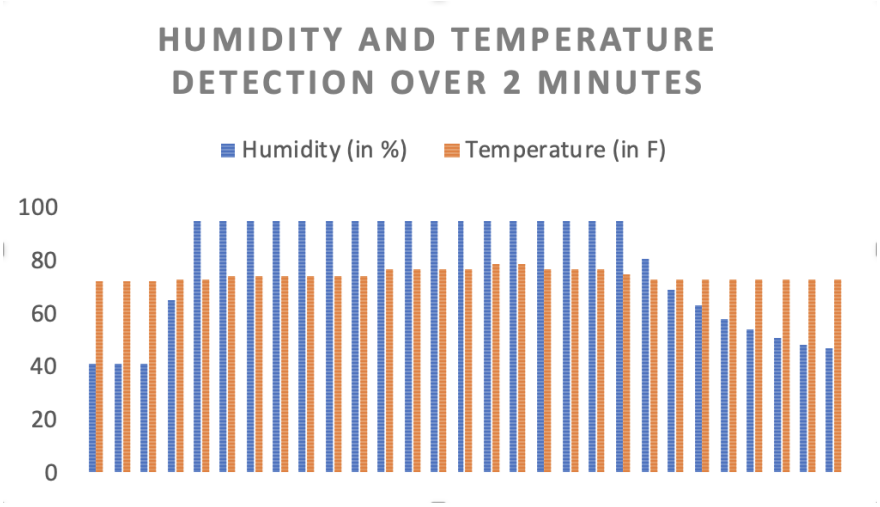

Figure 4.2: Humidity and temperature data for a working kettle — The DHT11 Sensor is placed directly above the sensor as can be seen in Figure Below. The humidity percentages reach 95% from 44% in less than 2 minutes while the kettle is boiling water. The temperature change stays relatively small as it reaches $77°F$, when the room temperature was $71°F$. The decline in humidity corresponds to the moment when we move the sensor away from the kettle's humidity.

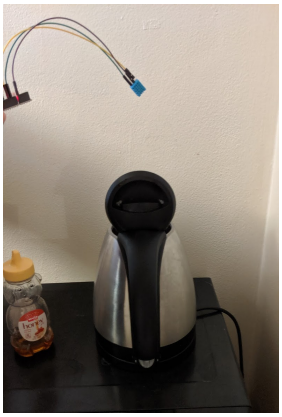

Figure 4.3: Humidity sensor over working kettle

Results: The results of this first experiment show us that different motion sensors may lead to different interactions. An IoT device that uses a PIR motion will not necessarily pick up on motion from other IoT device gestures. It also won't register any motion that is further than 33 feet [13]. The PIR sensor can only detect infrared changes in the environment. While, an ultrasonic motion device may or may not register a motion of the blinds within its range (13 feet), it is important to

note the error range for a commercial off-the-shelf sensor remains high. Last but not least, we also note a kettle can trigger a smoke alarm. The DHT11 temperature and humidity sensor we use has a small range for measuring the surrounding air, when we place the sensor within its range, right above the kettle, we observe an interaction.

Interaction Chain	Distance between two devices	Results
Motion of Blinds and PIR Motion Sensor	3 feet	No interaction
Motion of Blinds and PIR Motion Sensor	7 feet	No interaction
Motion of Blinds and Ultrasonic Motion Sensor	3 feet	No interaction
Motion of Blinds and Ultrasonic Motion Sensor	7 feet	No interaction
Kettle Humidity and Smoke Alarm	1 foot, devices placed right above each other	Interaction detected
Kettle Humidity and Smoke Alarm	3 feet, devices placed right above each other	No interaction
Kettle Humidity and Smoke Alarm	3 feet, devices placed horizontally	No interaction

Table 4.1: Experiment 1 interactions, measurements and results

4.2 Experiment 2

4.2.1 Experiment Description

Negative interaction: Security

The second experiment involves two separate application interactions. The first application comprises of a smart stove top. The smart stove top offers its user a safety reassurance: it can turn itself off in case it is left on accidentally. The stove is monitored by a connected motion sensor which checks for motion every 15 minutes to ensure there is still someone cooking on the stove. If no motion is detected, the smart stove turns off. While this seems like a great idea to prevent gas leaks or fire hazards, a second IoT application can adversely interact with this chain of events.

Our second application has the simple purpose of temperature control in the kitchen area. If temperature in the environment exceeds a certain threshold set by the user, it turns on the fan.

The stove top's sensor checking for motion can mistake the fan for a person cooking, while the indoor thermometer can pick up on the heat dissipated by the cook tops. The vicious cycle illustrated in Figure can be topped off with a gas leak alert or a smoke detector trigger depending on the type of the cook top (gas, electric, induction). Furthermore, as it was the case in the previous scenario, the triggered alarm can unlock and open all exits, leaving the user's home in a complete

vulnerable state.

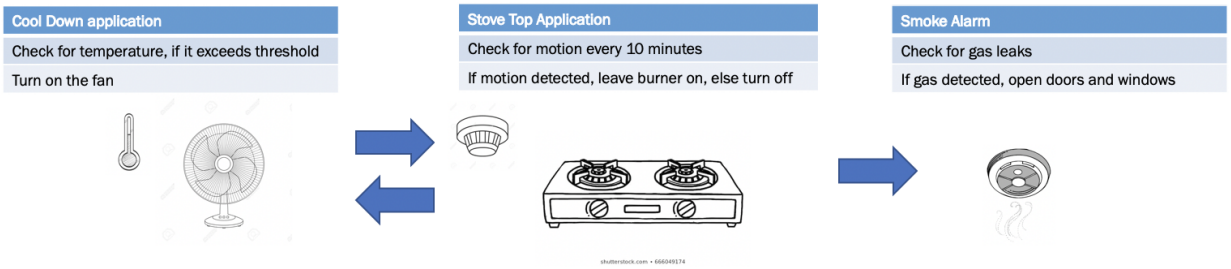

Figure 4.4: Experiment 2 Diagram

For safety purposes we didn't check if our smoke alarm detects any gas leaks. Nevertheless when we left the stove top on, we wanted to see the temperature change detection. When the board was 10 inches away from the stove, there were no temperature changes, the measurement stayed put at $70°F$ (see Figure 4.5). While the heat was unbearable for the researcher when the stove top was on at a high heat, the DHT11 didn't pick on the temperature change until it was placed 6 inches right above the burner, only then the DHT11 gave the output for $84°F$ after 8 minutes (see Figure 4.6 and 4.8).

Figure 4.5: DHT11 placed next to the burning stove top

Figure 4.6: DHT11 placed right above the burning stove top

4.2.2 Observations and Results

Observations: Contrary to the results of the first experiment, the PIR motion detector this time registered motion. One interesting finding to note is the changes detected are not from the fan's motion itself but, the infrared from the combination of a moving fan and a burning stove top. The two smart appliances when sitting close enough to each other create a warm air flow. The cold air blowing generated from the fan lets the heat travel and when every now and then the fan completes a turn, the PIR motion sensor records a change in the air flow temperature and registers it as movement.

When the same motion trigger was challenged with an Ultrasonic Sensor, movement was still detected. The reason behind the movement was due to the fan's changing angles, as Ding et al. expected in their research through NLP [3]. Figure 4.7 shows us that distance changes with every turn of the automated fan, and this is registered as movement.

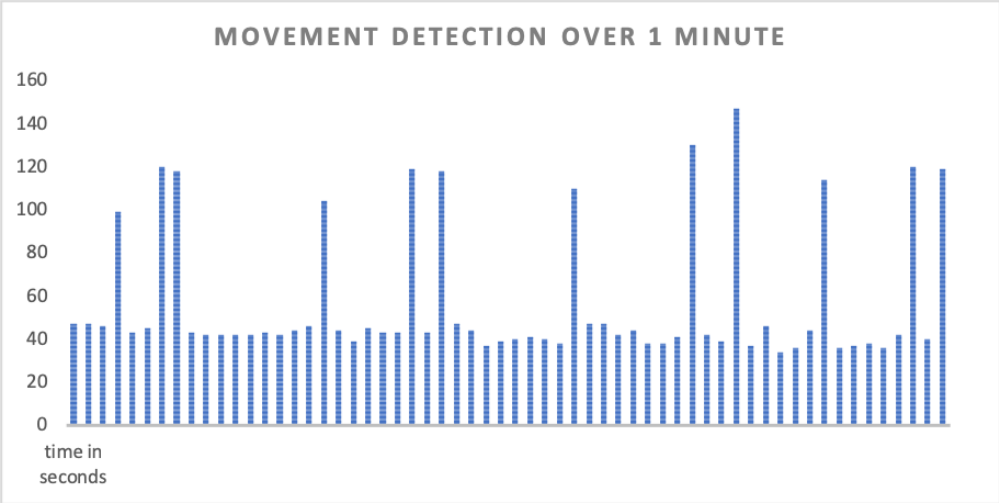

Figure 4.7: Ultrasonic Sensor detects fan motion — The ultrasonic waves detect the angle change of the fan, resulting in the spikes in the graph.

Finally, even though we couldn't test for any gas leaks, we observed the temperature change can only be detected if the sensor is placed close to the environment where it is supposed to monitor. While the high temperature pushed the resistance limits of the Arduino Board, the alarm we built was able to pick on the sudden temperature change. In this experiment, we were able to get the alarm to trigger.

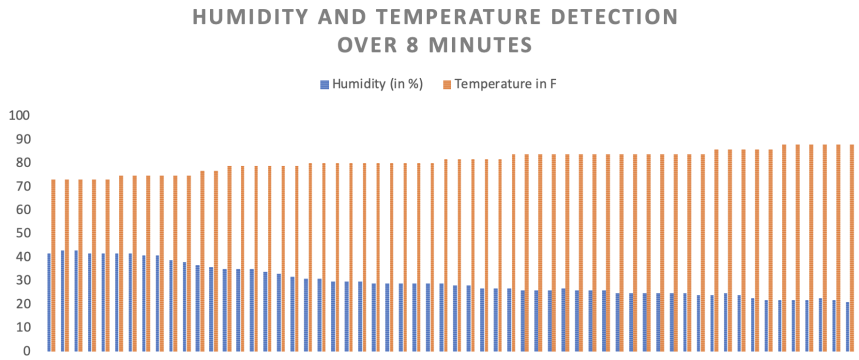

Figure 4.8: Temperature sensor data over working stove top — The DHT11 temperature and humidity sensor is placed 1 feet above the stove top that is already on, we can observe an increase from $70°F$ to $85°F$ over 8 minutes. We also see a decrease in humidity from 42 % to 22% because the stove top gets way too hot.

Results: Both motion sensors performed as expected, they both registered the fan's motion, resulting the stove top to stay on until a further outside event interrupts the cycle. Meanwhile, the alarm was only triggered if placed above the stove top. The results show that a user can be dangerously affected if they count on their smart stove to turn off. The stove will register the fan's motion and will not turn off until further action. This can result in serious consequences such as a gas leak, or a fire.

Interaction Chain	Distance between two devices	Results
Stove Top and Smoke Alarm	6 inches	Interaction detected
Stove Top and Smoke Alarm	10 inches	No interaction
Fan Motion and PIR Motion Sensor	3 feet	Interaction detected
Fan Motion and PIR Motion Sensor	10 feet	No interaction
Fan Motion and Ultrasonic Motion Sensor	3 feet	Interaction detected
Fan Motion and Ultrasonic Motion Sensor	10 feet	Interaction detected

Table 4.2: Experiment 2 interactions, measurements and results

4.3 Experiment 3

4.3.1 Experiment Description

Negative interaction: Privacy

The IoT device interaction depicted in this section is listed as the second high-risk interaction chain, classified by Ding et al. [3]. This final experiment is aimed to see the impact of low-cost and commercially off-the-shelf sensors, on negative interactions in the physical domain. Ding et al. mention that a motion sensor can be triggered by the motions of a blind and turn on lights. We will describe this experience with two different motion sensors.

Table 4: Top 10 High-Risk Interaction Chains

No.	Trigger1 Capability	Action1 Capability	Potential Device	Channel	Trigger2 Capability	Action2 Capability	Potential Device	Risk Score
1	locationMode	switch	heater	smoke	carbonMonoxide	locationMode	alarm(window, lock)	70.75
2	time	switch	feeder, curtain, fan	motion	motionSensor	locationMode	bulb, window	64.81

Figure 4.9: High Risk Interaction Chains by Ding et al. [3]

We would like to see if one simple application can trigger another humble one. One is designated to open the blinds in the bedroom at sunrise time while the user is still sleeping, enabling the user to wake up slowly with natural sunlight and improved circadian rhythm. The other application in the bedroom is programmed to turn on the lights if light bulbs sense any motion. Despite the fact that this interaction seems like no danger at first, it invades privacy by exposing a bedroom while the user is still asleep. IoT devices with bad sensing and not doing their jobs properly, generate security issues that need to be addressed.

As with the first experiment, we separately try two motion detectors attached to "light bulb" Arduino and see if they will register any change in movement.

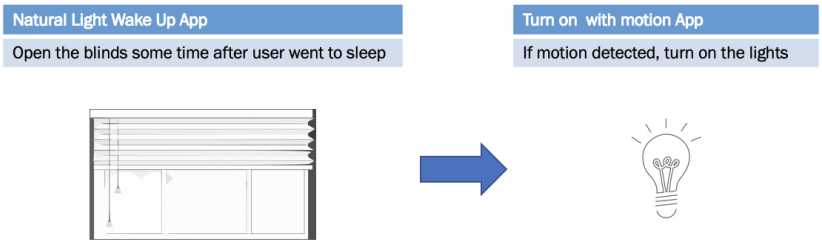

Figure 4.10: Experiment 3 Diagram

4.3.2 Observations and Results

Observations: The PIR motion detector does not pick on the motion of blinds, even when the motion detector is placed strategically right in front of moving blinds. This is because the blinds do not discharge any temperature changes. Unless the motion comes from a device that sends out infrared wavelengths, the light bulb attached to a PIR sensor will not get activated. Our second attempt for the experiment replaces PIR motion sensors with Ultrasonic motion detectors. The results confirm the findings of the first experiment, the change in distances recorded by the ultrasonic waves stay minimal, hence the blinds motion can not be registered by the motion detector. The light bulbs will not turn on with bad sensing.

Results: The results show that the interaction noted in Figure did not work as predicted by Ding et al., the interaction with curtains should not be classified as risky [3]. We conclude that it is not very likely to trigger a motion activated device with motion of the blinds. When we compare this experiment's results to the first experiment (Section 4.1), motion sensing is likely to happen only if there exists any resources that can cause a change in infrared waves.

Interaction Chain	Distance between two devices	Results
Motion of Blinds and PIR Motion Sensor	10 feet	No interaction
Motion of Blinds and PIR Motion Sensor	5 feet	No interaction
Motion of Blinds and Ultrasonic Motion Sensor	10 feet	No interaction
Motion of Blinds and Ultrasonic Motion Sensor	5 feet	No interaction

Table 4.3: Experiment 3 interactions, measurements and results

4.4 Addition to IoTBench

This work contributes to the IoT community by adding onto the IoTBench, scenarios and experiments in this work [4].

IoTBench is an open repository located on the software development platform Github. The repository consists of a collection of various IoT specific app analyses, enabling researchers, manufacturers and even users to ensure a safe experience with their IoT setups. The test corpus includes a range of security related findings from malicious apps, their methods and their analyses. Each application in the suite has warnings of the present data leaks, provided as comments in the source code.

As this project explored the security challenges posed by IoT devices, and was initially inspired by the founder of this repository, the negative interaction scenarios used in this work will be incorporated to the test suite. IoTBench currently has two categories for different IoT platforms, one for Samsung SmartThings Apps and another for OpenHab. The repository does not include an Arduino folder yet, and we are excited to contribute with a new section including the interactions listed in Table 4.4. This will allow a wider range of test evaluations for the IoT community.

Interaction Source	Triggered Device	Results
Smart Blinds	Motion Sensor	No interaction
Smart Kettle	Smoke Alarm	Negative interaction
Smart Fan and Smart Stove	Motion Sensor	Negative interaction
Smart Stove	Smoke Alarm	Negative Interaction

Table 4.4: List of IoT Device Interactions added to IoTBench

Chapter 5

Discussion

In Chapter 2, we introduced prior research findings and the techniques researchers have devised to detect risky IoT applications. In this work, we modeled IoT devices through replicating their most important features. It is important to note that the results in this paper apply to IoT devices that use cheap sensors. Furthermore, distances between IoT devices and their placements matter in overcoming the challenges listed in research papers. The modeled IoT devices are observed to interact with each other only within small ranges such as 13 feet (see Chapter 4.1). Therefore, we must emphasize the role of room sizes in overcoming negative interactions. Although room dimensions are generally bigger in the US compared to the rest of the world, having multiple IoT devices in the same room in places like Japan or big European cities increases the possibility of having physical interactions, as predicted in research.

Still, there remain many issues to be explored in future studies to allow end users an ideal IoT experience. Although it is hard to know which sources sensors react to, meaning the physical phenomena in device's environment, it is possible to further investigate the susceptibility of device triggers to interactions. This final chapter discusses the next steps security research should take in the IoT domain.

5.1 Investment in more accurate sensors

Although we concluded that some of the interactions which threaten user security can be avoided by using PIR motion sensors and ultrasonic sensors, the commercial, off-the-shelf sensors used in this work have some limitations. First, the range of sensors is very limited and usually cannot cover the entirety of a conventionally medium-sized room. Second, the ultrasonic sensor tested in Chapter 4.1 is promised to have an error rate of 3mm, but is actually much higher than that. The problem with motion sensing is not limited to off-the-shelf Arduino sensors only, as seen in Figure 5.1 . This review was taken from a Samsung SmartThings motion sensor comment section on Samsung's website [14]. The customer's complaint clearly implies the sensor requires more precision and well thought motion detection features.

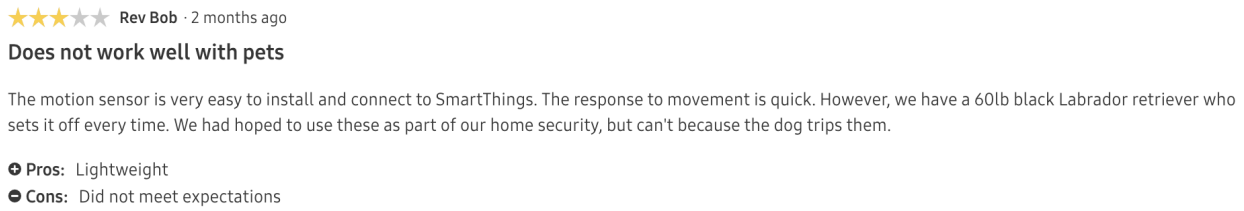

Figure 5.1: Samsung Motion Sensor Review

Finally, still remaining in the sensor realm, the DHT11 humidity and temperature sensor shows some limitations as the device only works when placed very close to the environment where the user expects the data collection to happen. Placing DHT11 in close proximity of measurement area gave expected results, however we know DHT11 is a low-cost sensor and it has a small range of temperature and humidity detection. In order to simulate an actual smoke alarm detector, a user would have to place multiple sensors everywhere around the house. Future research should focus more on improving the precision and widen the sensing range of both the commercial off-the-shelf sensors and the ones specifically crafted for advanced IoT devices.

5.2 Verification by sensors

While sensor reliability has been explored in the research world, sensors themselves in IoT devices have a high false alarm rate and remain vulnerable. This vulnerability is dangerous and poses a problem as it can lead to unaccounted failures, some demonstrated in this work.

One approach to increase the reliability of sensors would be the use of an array of sensors throughout the same environment. This array creates a network structure and analyzes the data collected from each sensor altogether. This way, the average of the collected data can be used to improve the robustness of the application [15].

Future studies should also focus on analyzing the environments in which the IoT devices are located. If the connected devices can establish an awareness of each other, false alarm triggers and negative interactions may be avoided. Usually, the phenomenon that triggers sensors in devices is hard to identify. However, developers should focus on a way to authenticate the source of the trigger. For instance a smoke alarm detector should be able to observe temperature differences and observe how quick temperature changes to possibly relate it to a nearby source. If an alarm is capable of noticing the connection between a temperature change due to the humidity and can detect a running, connected kettle, the first action to take could be to send an SMS to user's phone instead of automatically letting the alarm trigger and unlocking the doors.

Chapter 6

Conclusion

IoT has extended its reach from the industrial context to integrate itself in the everyday life of regular consumers. Internet-connected devices can cooperate, exchange information, and do the tasks assigned to them. As users become more comfortable installing IoT apps, and setting up multiple IoT devices in the same environment, the potential risks posed by interactions between devices will increase. Researchers have discovered that if multiple devices are used in the same environment, they can interact and cause scenarios harmful to the users. Scholars discovered IoT device interactions through analyzing IoT application analysis [2] [3]. In this paper, the theoretical findings of previous research were explored more in depth in the real world.

This work explored a way to model IoT devices and identify and observe applications classified as risky. First, we gave an introduction to IoT apps and their growing importance in our everyday lives. Then, background research was offered to emphasize the significant security dangers related to IoT apps. Finally, we investigated a way to model an experiment to confirm the possibility of negative interactions as depicted in research space. Devices can be modeled with different sensors and, and we discovered that the choice of sensors plays a significant role in the outcome of device interactions.

Overall, the findings of this work confirm that IoT device interactions happen in physical spaces and can result in scenarios that threaten the security, safety and of the consumer. In addition, the results indicate that it is important to carefully select the sensors when building an IoT device, and that it is just as essential to strategically place devices which share the same environment to prevent any unexpected interactions. Our experiments suggest that the IoT development and research communities should further broaden their IoT testing practices to include these concerns.

Appendix

```
#include <Servo.h>

#define servoPin 10

Servo servo;
bool first;

void setup() {
  servo.attach(servoPin);
  first = true;

}

void loop() {
  if (first==true){
      servo.write(0);
      delay(1000);
      servo.write(180);
      first = false;
      delay(1000);
  }
}
```

Algorithm 1: Timed Servo Motor

```
int pirMotionPin = 4;
int relayPin = 8;

void setup(){
  Serial.begin(9600);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);}

void loop(){

  while (digitalRead(pirMotionPin) == HIGH) {
       digitalWrite(relayPin, HIGH);
       Serial.println("Relay is on");
       delay(5000);
}

       digitalWrite(relayPin, LOW);
       Serial.println("Relay is off");
       delay(5000);
}
```

Algorithm 2: PIR Motion Sensor activated Power Relay

```
#define outputPin 6
#define inputPin 5
#define relayPin 9

void setup(){

Serial.begin (9600);
pinMode(outputPin, OUTPUT);
pinMode(inputPin, INPUT);
pinMode (relayPin, OUTPUT);}

void loop(){

int duration, distance;
digitalWrite(outputPin, HIGH);
delayMicroseconds(5000);
digitalWrite(outputPin, LOW);

duration = pulseIn(inputPin, HIGH);

distance = (duration/2);

if (distance < 40){
relayOn();}

Serial.print(distance);
Serial.println(" cm");
delay(5000);
}

void relayOn(){
digitalWrite(relayPin, HIGH);
delay (5000);
digitalWrite(relayPin, LOW);}
```

Algorithm 3: Ultrasonic Motion Sensor activated Power Relay

```

#include <Adafruit_Sensor.h>
#include <DHT.h>
int buzzerPin = 8;

#define DHTPIN 5
#define DHTTYPE DHT11
#define LED_TOO_HOT A2
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  pinMode(buzzerPin, OUTPUT);
  pinMode (A0 , OUTPUT);
  pinMode (A1 , OUTPUT);
  pinMode (A2 , OUTPUT);
  delay(50000);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  float f = dht.readTemperature(true);

  Serial.print("Humidity: ");
  Serial.print(humidity);
```

```
  Serial.print("Temperature: ");
  Serial.print(temperature);

  if (t >= 30) {
    digitalWrite(A2, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay (50000);
    digitalWrite(buzzerPin, LOW);
    digitalWrite(A2, LOW);}}
```

Algorithm 4: DHT11 Temperature Sensor and Buzzer

```
#include <Adafruit_Sensor.h>
#include <DHT.h>

#define relayPin 13
#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin (9600);
  pinMode(relayPin, OUTPUT);

  dht.begin();
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  float f = dht.readTemperature(true);

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("Temperature in C: ");
  Serial.print(temperature);

  if(temp > 20){
    digitalWrite(relayPin, LOW);}
    else{
    digitalWrite(relayPin, HIGH)}
}
```

Algorithm 5: DHT11 Temperature Sensor activated Power Relay

Bibliography

[1] Egham. Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. {https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016}, Feb 2017.

[2] Z Berkay Celik, Patrick McDaniel, and Gang Tan. Soteria: Automated iot safety and security analysis. In *2018 USENIX Annual Technical Conference*, pages 147–158, 2018.

[3] Wenbo Ding and Hongxin Hu. On the safety of iot device physical interaction control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 832–846. ACM, 2018.

[4] Z. Berkay Celik, Leonardo Babun, Amit K. Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. Sensitive information tracking in commodity iot. In *USENIX Security Symposium*, Baltimore, MD, August 2018.

[5] DataFlair Team. How iot works - 4 main components of iot system. https://data-flair.training/blogs/how-iot-works/, Sep 2018.

[6] Nicole Casal Moore. Hacking into homes: 'smart home' security flaws found in popular system. https://news.umich.edu/hacking-into-homes-smart-home-security-flaws-found-in-popular-system/, May 2016.

[7] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 636–654. IEEE, 2016.

[8] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. Sensitive information tracking in commodity iot. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1687–1704, 2018.

[9] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117. ACM, 2017.

[10] Last Minute Engineers. Home. https://lastminuteengineers.com/, Apr 2019.

[11] Adafruit Industries. Adafruit power relay featherwing. https://www.adafruit.com/product/3191.

[12] Will a smoke detector detect heat? https://www.alarmgrid.com/faq/will-a-smoke-detector-detect-heat.

[13] What is a pir motion sensor: Pir ic working, features and applications. https://www.elprocus.com/pir-sensor-basics-applications/, Mar 2016.

[14] Samsung smartthings motion sensor (2018) smartthings - gp-u999sjvlbaa: Samsung us. https://www.samsung.com/us/smart-home/smartthings/sensors/samsung-smartthings-motion-sensor-2018-gp-u999sjvlbaa/.

[15] Piero Zappi, Elisabetta Farella, and Luca Benini. Enhancing the spatial resolution of presence detection in a pir based wireless surveillance network. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 295–300. IEEE, 2007.
url

Academic Vita

ELIF ERDOGDU

linkedin.com/eliferdogdu, eliferdogdu1@gmail.com, (814)321-8101, State College PA

EDUCATION

The Pennsylvania State University, University Park *Aug 2015 - Dec 2019*
Schreyer Honors College
Bachelor of Science in Computer Science

PROFESSIONAL EXPERIENCE

Penn State University State College, PA
Computer Security Research Assistant *Nov 2018 - Nov 2019*

- Demonstrated security challenges and unsafe environments in of IoT applications.
- Researching further ways to overcome unsafe interactions between connected devices by authenticating action triggers.

Netas Telecommunications Inc Ankara
Software Eng. Intern *Jul 2018 - Aug 2018*

- Created operation flowcharts of the client enterprise for modules after running 10+ interviews with client staff,
- Designed 30+ module implementation instructions into clients crisis management software,
- Documented 40+ software module codes into flow charts to present to the client with JasperReport.

ING Bank Istanbul
Tech Consulting Intern *Jul 2017 - Aug 2017*

- Created a web-based banking platform module. Converted offline FINSOFT software using C and ASP.Net,
- Prepared white papers on "Use of AI in chat bots for better customer experience." Researched DevOps platforms, machine and deep learning algorithms.

SOCIAL INVOLVEMENT

Leadership Experience

- Taught 6th and 7th graders about Machine Learning at Penn States Girls Camp for Technology,
- Selected Scholar to advocate for affordable tuition at the Capital Day 2016 in Harrisburg, PA,
- Raised $500 for State College House of Care non-profit. Co-organized fundraising dinner for 50 attendees.

Penn State French Society

- Key communicator between the executive board and 200+ members,
- Increased meetings experience by presenting weekly to a group of 30+ members. Leveraged fun activities and lunch & learns enabling students to build strong relationships.

SKILLS

Coding Skills	C/C++, Java, Python, HTML, SQL, Verilog, ASP.NET, Latex.
Languages	French, Turkish, English. Limited proficiency in German.
Awards	Schreyer Academic Excellence Award, Deans List, French Baccalaureate with Honors.