Turning Legacy IR Devices into Smart IoT Devices

Chuta Sano[1], Chao Gao[1(✉)], Zupei Li[1], Zhen Ling[2], and Xinwen Fu[3]

[1] University of Massachusetts Lowell, Lowell, MA 01854, USA
{schuta, cgao, zlil}@cs.uml.edu
[2] Southeast University, Nanjing, China
zhenling@seu.edu.cn
[3] University of Central Florida, Orlando, FL 32816, USA
xinwenfu@cs.ucf.edu

Abstract. In this paper, we introduce a low-cost setup to convert an infrared (IR) controllable device to a smart IoT device. We design and implement a circuit board containing an IR sensor and multiple IR LEDs in parallel that transmit up to around *seventeen meters*. The board has an interface that can be connected to a Raspberry Pi and other similar devices. Our IR signal learning tool can record IR signals of an IR remote while our IR replay tool can replay the recorded signals to control the corresponding IR device. Therefore, a smartphone can be used to remotely control IR devices through an MQTT broker on a cloud server since the Raspberry Pi can be connected to the cloud. We also introduce the security implications of infrared communication using our setup and demonstrate attack scenarios. For example, a drone armed with our device can remotely turn off a TV - https://youtu.be/rPbzPbWrbf8 or http://v.youku.com/v_show/id_XMzQ0Njc5MzM3Ng.

Keywords: Internet of Things · Infrared · Smart home · Drone

1 Introduction

The Internet of Things (IoT) is a world-wide network of uniquely addressable interconnected objects [1, 2]. Many household devices including TVs, fans, air conditioners and toys have IR receivers and are controlled through IR remote. We may want to connect these legacy IR devices to the Internet and control them remotely while maintaining an acceptable cost level.

In this paper, we aim to design a cheap bridge between IR devices and Internet. In other words, we want to implement a smart IR remote with the Internet capability. To this end, we design a custom IR signal recording and replay circuit board, which can be connected to a Raspberry Pi and similar low-cost single board computers. For example, the new Raspberry Pi Zero W with the WiFi capability costs only $10 dollars. Without loss of generality, we will use the popular Raspberry Pi as an example in this paper to demonstrate how our IR board is used. We implement an IR recording tool that can record IR signals from an IR remote of any IR device. Our IR replay tool can replay the signal and control the IR device. Therefore, the Raspberry Pi can be connected to the

S. Chellappan et al. (Eds.): WASA 2018, LNCS 10874, pp. 412–424, 2018.
https://doi.org/10.1007/978-3-319-94268-1_34

Internet and a smartphone can be used to remotely control the IR device through the Raspberry Pi. The smartphone and Raspberry Pi can be interconnected with a IoT broker on the cloud such as Amazon EC2.

The contributions of this paper are summarized as follows. We introduce a low-cost and extendable model to transform IR remote controllable devices into smart IoT devices. To the best of our knowledge, this work is the first to fully address IR playback through both hardware and software. An IR transceiver module is made for the Raspberry Pi along with software. We also introduce the security implications of infrared communication using our setup and demonstrate attack scenarios. For example, we demonstrate that a remote-controlled drone armed with our device can turn off/on a TV. Please refer to the YouTube video https://youtu.be/rPbzPbWrbf8 or YouKu video http://v.youku.com/v_show/id_XMzQ0Njc5MzM3Ng.

The rest of this paper is organized as follows. We introduce background knowledge including infrared communications, Raspberry Pi and Message Queuing Telemetry Transport (MQTT) in Sect. 2. The hardware and software of the IR board is elaborated in Sect. 3. We discuss the security implications of the device in Sect. 4 and evaluate the board in Sect. 5. Related work is introduced in Sect. 6 and we conclude the paper in Sect. 7.

2 Background

2.1 Infrared Communications

Infrared communications start with the sender rapidly turning pulses into a series of 940 nm wavelength electromagnetic waves, usually via LEDs, which the receiver decodes into bits based on a scheme. The sender modulates the pulses at usually 38 kHz and encodes the signal to reduce noise and jitter, such as from the sun, while increasing accuracy of transmission.

Figure 1 shows the three well known forms of encoding a signal: *pulse distance*, *pulse length* and *bi-phase*. In this subsection, encoder refers to the sender, decoder refers to the receiver, "on" refers to the infrared source (e.g., an LED) being on, and "off" refers to the infrared source being off.

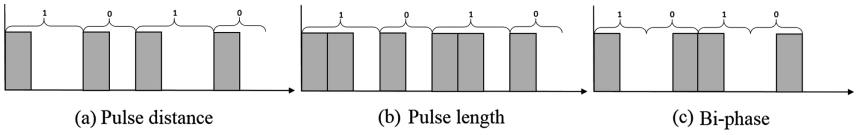

Fig. 1. IR signal encoding

The pulse distance encoding scheme takes a constant on-length and two different off-lengths for a 1 and a 0. It encodes a binary 1 with an *on* for on-length microseconds and then an *off* for off-length-1 ms and likewise, a binary 0 with an on for on-length microseconds and then an off for off-length-0 ms. Note that the on duration is fixed for both binary 1 and 0 and the off duration determines whether it is interpreted as a 1 or 0.

The pulse length encoding scheme, similar to pulse distance, takes two different on-lengths for a 1 and a 0 and an off-length; it encodes a binary 1 by a sequence of on for on-length-1 ms then an off for off-length and likewise, a binary 0 with an on for on-length-0 ms and then an off for off-length microseconds. In this case, the off duration for both binary 1 and 0 is fixed and the on duration determines whether the signal is detected as a 1 or a 0.

The bi-phase encoding scheme takes a timeslot, also known as time window, and encodes a binary 1 by a sequence of *on* → *off* and a binary 0 by a sequence of off → on. The timeslot determines how long each off and on would be; it is a constant that must be exchanged to both the sender and the receiver beforehand. Unfortunately, bi-phase encoding may potentially cause decoding to be flipped; if the decoder does not detect the start of a bi-phase and misses an odd multiple of timeslots, it can incorrectly detect a binary 1 as a 0 and vice versa. Figure 2 is an example of bi-phase decoding flip. The sequence 10110 is encoded but because the receiver misses the first on pulse, it incorrectly decodes the sequence as 01001. Therefore, in practice, a known header is implemented to guarantee that the decoder and encoder are in sync, for example with the RC5 protocol.

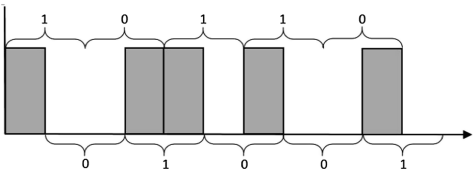

Fig. 2. Bi-phase decoding flip

Many manufacturers follow a variation of the RC5 or the NEC protocols, which are both modulated at 38 kHz. The RC5 protocol uses the bi-phase encoding scheme with the time slot being 1.78 ms and consists of a 2-bit header "11", an alternating bit, 5 address bits, and 6 command bits. The alternating bit alternates on retransmission within a button press, for example if a button is continuously pressed [3]. RC5 is most commonly used by American and European manufactured audio and video equipment such as speakers.

The NEC protocol uses the pulse distance encoding scheme with its on-length being 562.5 ms, the off-length for 0 being 1687.5 ms, and the off-length for 1 being 562.5 ms. The NEC protocol starts with a header of 9 ms on and 4.5 ms off. Then, it sends 8 address bits followed by its inverted form and then 8 data bits followed by its inverted form for a total of 32 bits. For example, the full body of a command with an address of 01001000 and data of 00000001 will be 01001000, 10110111 (inverted address), 00000001, 11111110 (inverted data). Finally, it sends an additional 562.5 ms of on as a footer. In cases of a repeat, for example when the volume button is continuously pressed, it waits for approximately 40 ms, sends a 9 ms on, a 2.25 ms off, and finally a 562.5 ms on [4].

A problem with emulating infrared communications on an OS like Linux is that the protocols are sensitive to time, and using the system clock as a form of time tracking is not necessarily precise enough. Furthermore, since processes share time with other processes, the infrared communication process cannot run accurately enough.

2.2 Raspberry Pi and PiGPIO Library

Raspberry Pi is a lightweight computer that runs on an ARM CPU. Although various OSes are compatible with Raspberry Pi, in this paper, all Raspberry Pis use Raspbian, which is a Debian-based Linux OS. In this work, Raspberry Pi 2 and 3 were used, but below we establish common features between other models to emphasize the point that any Raspberry Pi and other similar devices can be used.

The PiGPIO library is a GPIO interface written in C that can handle time sensitive GPIO tasks by running a helper daemon. Its primary use in this project was to generated time accurate pulse waves. Using traditional file I/O methods to send pulses was not precise enough due to various delays introduced by the overhead (e.g., I/O and time-sharing with other processes).

2.3 MQTT

Message Queue Telemetry Transport (MQTT) is a popular protocol to implement IoT communications due to its lightweight and simple nature. MQTT is a topic-based publish/subscribe messaging system. A topic is a unique string that serves as the identifier for a type of message. A publisher is any client that sends messages, which contain the topic and the payload, whereas a subscriber is any client that listens for incoming messages from some topic. A client or node is any system that connects to a broker and publishes and/or subscribes to topics.

Mosquitto [5] is an open source implementation of MQTT 3.1 including MQTT over TLS. Mosquitto provides a broker executable along with publish and subscription tools. Paho-mqtt is a library for MQTT that provides APIs to subscribe and publish to an MQTT broker.

3 IR Transceiver and Smart IR Devices

In this section, we first introduce the hardware and software of our long-range IR transceiver. We then briefly discuss how to convert legacy IR devices into smarts one with our IR transceiver so that we can control it from the Internet anywhere.

3.1 Hardware

Figure 3 illustrates the schematic of our IR transceiver board, Fig. 4 shows the PCB design and Table 1 lists its parts. A typical usage setup will consist of a Raspberry Pi and a transceiver module. The transceiver uses four GPIO interface to connect the Raspberry Pi. It receives power from two GPIO pins, a 5 V pin, and a ground pin from the Raspberry Pi. The sender GPIO pin (J-1) receives signals from the Raspberry Pi to

turning on and off the LEDs on and off respectively and the timing is controlled by the Raspberry Pi. The receiver pin (J-2) of the transceiver connects to the Raspberry Pi which records the signals.

The transmitter portion of the transceiver has three important improvements in design compared with related work [6]. First, PNP (Q1) transistors are used to pull a 5 V power pin instead of directly drawing from the GPIO pin which only offers 3.3 V power. This increases the intensity of the light, leading to higher range of IR transmission. Second, power supply is routed through a 220 uF capacitor (C2) and a 0.1 uF capacitor (C1). Because the infrared protocols are fired in short bursts, the 220 uF capacitor can store charge while the LED is off to increase stability in cases where the Raspberry Pi fails to transmit steady current, and the 0.1 uF capacitor removes small electric noise. Finally, four IR LEDs (D1–D4) are placed in parallel to maximize coverage. the two outer LEDs are wide and short-ranged, whereas the two inner LEDs are narrow and long-ranged. The GPIO command is passed through the PNP transistor to each NPN transistor (Q2–Q5), which draws power from the 5 V pin in parallel. This design allows a range of approximately 10.0 m compared to a trivial GPIO-resistor-LED design which reaches approximately 2.0 m.

Table 1. Parts list

ID on PCB design	Part name
C1	Ceramic 0.1 uF capacitor (COM-08375)
C2	220 uF capacitor with 5 V + rating
D1, D3	Wide IR LED (IR333C/H0/L10)
D2, D4	Narrow IR LED (IR333-A)
Q1	PNP transistor (PN2907)
Q2, Q3, Q4, Q5	NPN transistor (PN2222)
TSSP58038	38 K IR receiver module
R1	1 KΩ 1/4 W 5% resistor

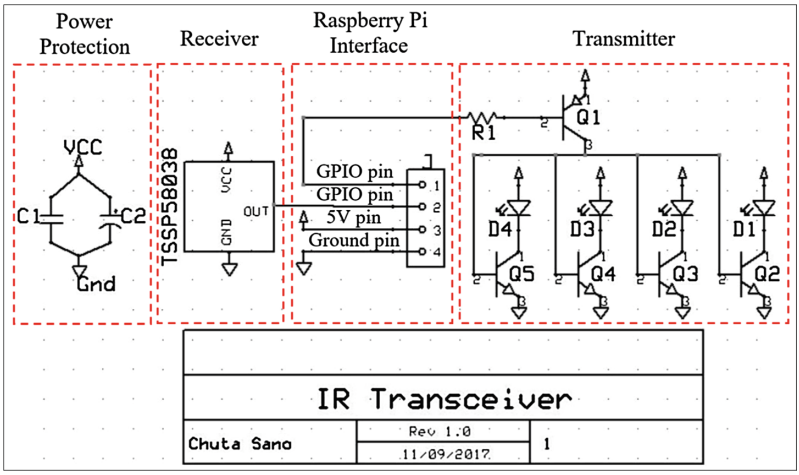

Fig. 3. Schematic of the IR transceiver board

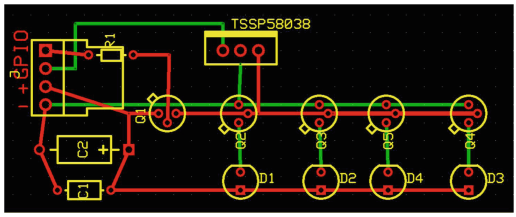

Fig. 4. PCB design of the IR transceiver board

The receiver module uses an integrated 38 kHz module of 940 nm peak wavelength, which can automatically filter out infrared light that is not modulated at 38 kHz. This decision is made for higher stability and distance compared to a generic light sensor and is justified because consumer IR protocols mostly use 38 kHz modulation [7].

3.2 Software

Separate software to interface with the sender and the receiver is designed along with a lightweight library that abstractifies much of the low-level intricacies in infrared communication.

The receiver software, written in C++, records the durations of low and high inputs from the receiver and records them into a raw format consisting of positive numbers representing the duration in microseconds to turn on the LED and negative numbers representing the duration in microseconds to turn off the LED. A user can associate a name to a recording. The recording is saved to "ircodes.txt" by default but can be optionally specified in the receiver software's first argument. The saved recording consists of a text file with two lines; the first line contains "name: <name>" where <name> refers to the chosen name in the prompt from the receiver. The second line consists of a list of integers delimited by space. A positive integer refers to turning on the LED for that many microseconds, and a negative integer refers to turning off the LED for that many microseconds. Additional recordings can be concatenated into the same file.

The sender software uses the PiGPIO library for nanosecond level accuracy in GPIO control, which is necessary to correctly replay signals. The software takes the name of the command as the first argument, matches it to the line reading "name: <name>" and as described previously, replays the list of numbers accordingly. The sender software attempts to read from "ircodes.txt" by default but the file name can be optionally specified in the second argument.

We have also written an extensive C++ library that abstractifies the sending of IR signals using our long-range IR transceiver. The sender library implements the previously mentioned raw format scheme, pulse-distance, pulse-length, and bi-phase encodings, and the NEC and RC5 protocols.

3.3 Smart IR Devices

Using the IR transceiver hardware and software, we can now connect our long-range IR transceiver to a Raspberry Pi and control an IR device by recording and replaying its IR signals. Since a Raspberry Pi has the Internet capability, we can connect the Raspberry Pi to the Internet and control the IR device from anywhere.

We give an example setup of controlling an IR device from the Internet. We set up a Mosquitto broker on an Amazon EC2 server so that we can publish messages through the smartphone. A Raspberry Pi can subscribe to a topic and send the appropriate signal upon receival of a message to the connected IR device.

4 Security Implications

In this section, we discuss the threats of the long-range IR transceiver that may be used to attack IR controllable devices of various kinds.

4.1 Replay Attack

Similar devices (from the same brands) do not change their IR codes on a per device basis, so following a simple replay procedure like introduced in the previous sections can allow anyone to be able to control the same devices. TVBGone follows a similar approach where the authors pre-record various brands of TV on/off toggle signals and replay it. With TVBGone, replaying all its code take approximately 2 min, which is certainly a very reasonable time frame to attack any TV [6]. Our long-range IR transceiver is controlled by a Raspberry Pi and can be easily customized to attack any IR controllable device.

4.2 Brute-Force Attack

The brute-force attack traverses all possible bits of a given protocol. For NEC, the set of all possible commands are based on unique address and command bits, both being 8 bits long, for a total of 16 bits or 2 bytes. For RC5, there are 5 address bits and 6 command bits for a total of 11 bits of entropy. A simple ascending approach where the address bit and command bit are considered one number and incremented was implemented. For example, the brute-force algorithm for NEC would start at 0x0000 (address = 0x00 and command = 0x00), and then try 0x0001 (address = 0x00 and command = 0x01), and so on until 0xFFFF. Generally, small items and products made in Asia use the NEC protocol.

As an example of the brute-force attack, a remote-controlled light strip, "BINZET 5 M 50 LEDs 3AA Battery Operated Copper Wire String Light LED Fairy Light LED Starry Light Cool White Festival Accent Light with Remote Control," was attacked via NEC brute-force, the objective being to turn on the light strip. The starting bits for the address and command bits were set to 0x00 and 0x00 respectively; the light strip was brute-force attacked within milliseconds, which was expected because the light strip's actual command was NEC with address 0x00 and command 0x02.

The simple incremental approach that was implemented as mentioned above traverses through all possible commands in one minute and twenty seconds per address. Therefore, this approach is estimated to take up to five hours and forty minutes in the worst case. However, in a fully optimized scenario, which can be implemented by creating a kernel module that accepts queueing of pulses to send (meaning subsequent commands are sent without delay), each command takes 67.5 ms to send, and there are 65536 distinct commands; the worst-case scenario can be shorted to just over one hour. Due to lack of feedback (the automated system cannot detect whether the right code was sent or not), the worst case must be considered over the average case.

An RC5 brute-force attack can be implemented the same way; the RC5 protocol, which only has eleven varying bits and takes 24.9 ms to send, can be completely brute-force attacked within fifty-one seconds.

An interesting observation was that the light strip accepted three command signals within the address 0x00. The timings at which the incorrect commands were accepted were varied and unpredictable; the reason for this unpredictable behavior remains unclear and may be a hardware issue on the lightstrip.

4.3 Drone Attack

One difficulty of attacking IR devices is the attacking IR transceiver has to be close to the target IR devices. To circumvent accessibility issues, we can put the Raspberry Pi and our IR transceiver on a drone. The Raspberry Pi can be connected to the Internet so that we can control the IR transceiver from anywhere.

Figure 5 shows an example of drone attack with a DJI Phantom 2. To minimize weight and therefore maximize flight duration, the on-board camera was removed, and a Raspberry Pi 2 with Anker PowerCore + Mini 3350 mAh were mounted. The battery powers the Raspberry Pi 2. The drone flew outside a second-floor conference room. This attack shows a method for attackers to partially circumvent the line of sight requirement; if a line of sight exists from outside through a window, then a drone can be leveraged to provide line of sight for the infrared setup.

Fig. 5. Drone attack against a TV through a window

5 Evaluation

In this section we introduce the device setup in terms of hardware and software. We estimate the cost of the entire setup, evaluate our hardware, software, as well as the attacks on infrared communication, and then discuss limitations and potential improvements to the system.

5.1 IR Device Setup

Figure 6 shows the smart IR device test setup. On the left side is an IR controlled LED (wrapped around a tree-like artifact) with its IR receiver. On the right side, the IR transceiver circuit board is connected to the Raspberry Pi Zero W [8] through GPIO pins. The four connected pins from bottom to top on the IR transceiver are connected to GPIO 22 (sender pin), GPIO 23 (receiver pin), a 5 V power pin, and a ground pin on Raspberry Pi Zero W, respectively. The Raspberry Pi Zero W is connected to a monitor, a power supply and a keyboard and mouse dongle through a USB OTG cable.

5.2 Cost

The total cost of the IR transceiver module is approximately $5.00 due to the variability of PCB printing (for example building one module is very expensive due to PCB printing). This experiment was done on the Raspberry Pi Zero W, which costs only $10.00. The software is computationally trivial, more than enough for a Raspberry Pi Zero W. Therefore, the total cost of the entire setup can be minimized to approximately less than $15.00 with a Raspberry Pi Zero W.

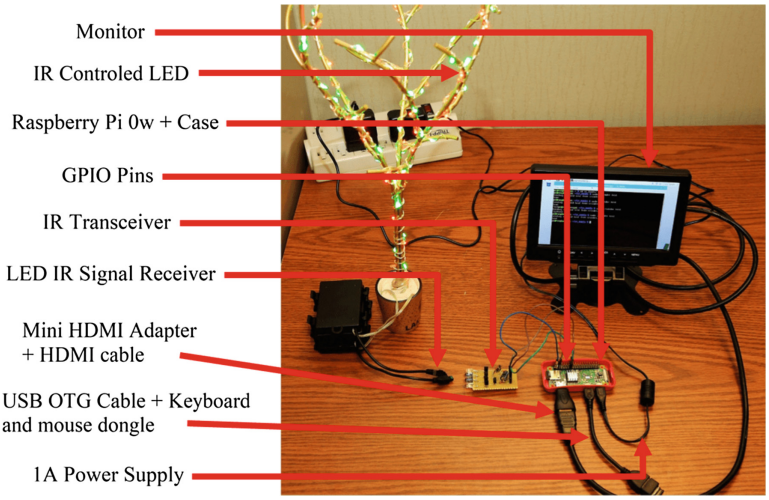

Fig. 6. IR device test setup

5.3 Range of Coverage

The range of the signals depends on multiple factors of the environment. High reflectivity of walls, narrow space, and minimal external noise (e.g., minimizing lights from sun or incandescent light bulbs) all increase the coverage and range of the signal. A similar approach to Acoustic Theory can be taken to fully describe IR transmission in rooms. Another huge factor is the sensitivity and noise tolerance of the IR sensor in the device: a higher quality IR sensor (higher sensitivity and better noise filtering) will also increase the range of the signal.

Table 2. Maximum distance for given situations

Situation	Maximum distance (m)
Inside, a building corridor	38.00
Inside, 13 × 13 m room with no pockets	17.70
Outside, through a window, receiver inside a room	13.00
Outside, receiver facing the sun	1.77
Outside, receiver facing away from the sun	4.26
Outside, in shade	5.83

Table 2 illustrates the maximum range of the IR transceiver module for indoor and outdoor use. The maximum range was determined by finding the highest distance between our transceiver and the aforementioned lightstrip module such that a signal was responsive ten out of ten times. As expected, the distance was maximized inside where there are likely minimal noise. Moreover, windows can decrease the range, mainly because of its reflectivity. Therefore, to perform a drone attack presented in Sect. 4.3, the drone should be at most 13 m away from the target device, and in most cases the drone can be very close (within 0.5 m) to the window, meaning target devices that are at most 12.5 m away from windows are vulnerable. All three outside cases show that the bounce of the light is very important for high distance. Furthermore, the experiment showed that the receiver tends to be impacted by the sun more than the sender; the sender side is not affected as much as shown by its maximum distance being similar to the Outside, in shade case.

The software was mainly developed as a usability improvement to the Linux Infrared Remote Control (LIRC) package. Although LIRC is a powerful tool, its usability is lacking for simple signal replaying purposes, where attempting to learn a signal not only failed at very noticeable rates but also took approximately thirty seconds. The entire recording process of our software is very quick and does not fail. Furthermore, the software implements various APIs to abstractify GPIO interaction, simplifying any generic infrared communication needs.

5.4 Security Analysis

Although it is clear that infrared communication has no security features and therefore suffers from replay attacks and brute-force attacks, lack of accessibility and impact are major problems that need to be addressed to make attacking infrared worthwhile.

Lack of accessibility comes from two factors: distance and blockage. Distance of infrared light is limited because as previously described, there are too many sources of potential interference, meaning infrared light becomes increasingly unstable with longer range. Blockage comes from various sources that block infrared light, such as walls. A possible workaround using drones was proposed.

The drone attack demonstrated that any IR devices with line of sight, for example through windows, to target devices can be exploited. To a determined attacker, using a similar attack alleviates some accessibility concerns. A potential issue with drone attack is its motor noise may attract the attention of the target. A drone's endurance (flight time per charge) may also affect the effectiveness of the attack.

We now discuss the impact of the attack. Turning off someone's TV is not as significant of an issue as stealing someone's credit card information. However, IR controllable air conditioners, fans, and thermal control units, which are popular in Asian countries, are certainly devices that need to be kept secure. Exploiting those devices to arbitrarily change room temperatures can cause health risks but also a non-trivial amount of monetary damage. Even controlling a TV can cause significant harm if an attacker were to turn it on and maximize the volume at night.

5.5 Limitations

Because of the nature of the infrared spectrum, it behaves almost identical to the visible spectrum. We essentially need a line of sight between the transceiver and the device to perform any IR communication. We worked on addressing this issue by maximizing both the distance and coverage of the IR module. At shorter distances, line of sight is not necessary because the strong and spread signal will bounce off other surfaces and reach the destination. However, this issue implies that multiple setups may be needed depending on the layout of the house and where the legacy IR devices are located.

Another problem is that there is lack of feedback due to the one-way communication of infrared controlled devices. For example, users receive feedback from turning on their TV by visually seeing the television turn on; they know to press the power button again if they visually see that the TV does not turn on in a reasonable timeframe. However, turning on the TV through Internet means that a user cannot get feedback about the TV's status unless the user is already within reasonable range from the TV, which would make the added connectivity meaningless. Therefore, extra sensors, such as camera or mic must be added, and those sensor inputs must be trained on a per device basis to fully consider the legacy device as IoT. Nevertheless, in an indoor scenario, interference is rare because of the high intensity of the emitted lights from the proposed design, meaning an infrared equivalent of packet loss is rarely an issue assuming the placement of the transceiver module is not too far, so users can assume with high confidence that any sent signal will be received.

6 Related Works

Overall, we were not able to find any published works that fully covered both the hardware and software aspects of IR playback introduced in this paper.

The Linux Infrared Remote Control (LIRC) is a software package for Linux that abstractifies much of the low-level details in infrared communication. It runs a helper daemon on the kernel level, which allows precise timing when sending infrared commands [9]. Although LIRC is very powerful, its recording process is very tedious, taking close to a minute to learn simple commands and sometimes even failing. Our software is a usability improvement to the LIRC.

TVBGone is a lightweight module that sends pre-trained TV on/off signals. It is controlled by an IC chip, and precise timing is obtained through a ceramic resonator. Its circuit design contained many clever tricks to increase stability and range in the transmission, which this project heavily drew upon (see Sect. 2.1). Its purpose is to turn off any TV, and it does so by hardcoding pre-recorded TV power toggle signals in the IC chip [6]. Therefore, it is different in design from our work which aims to record and playback any signal.

IrSlinger is a simple infrared sender which uses the PiGPIO daemon for precise timing. It uses a PN2222 transistor to route 5 V current to the LED instead of using the 3.3 V from GPIO [10]. However, it is potentially unstable because of the possibly unstable current from Raspberry Pi, and its coverage is not too large because it only uses one LED. Furthermore, the software does not fully automate playback, and users need to manually record and hardcode recorded values into a program.

Arduino Universal Remote autodetects NEC, RC5, and RC6 to handle for their repeat codes. It can only record and play back one signal. In our work, repeat codes were not handled at all because through testing we were not able to find any devices that were not responsive to sending the same code.

IRRemberizer uses a IR photo transistor to record signals instead of a IR transceiver, which allows IRRememberizer to record modulations of 30 k to 60 kHz at the cost of increased unreliability and therefore decreased range due to the photo resistor more likely to be affected by interferences from external infrared sources [11]. In this paper, a 38 kHz IR receiver module was used instead, which allowed higher stability and range in the recording process.

7 Conclusion

In this paper, we introduce a cost-effective method to transform legacy IR controlled devices into Internet connected smart IoT devices through a Raspberry Pi with an IR transceiver module. The cost of the IR transceiver is around $5.00. Our IR transceiver design achieves long-range coverage. We also introduce an extensive IR communication library to show security flaws in common IR protocols. A drone armed with our device may pose severe threats against IR controllable devices of various kinds.

Acknowledgments. This work was supported in part by US NSF grants 1461060, 1642124, and 1547428, by National Science Foundation of China under grants 61502100 and 61532013, by Jiangsu Provincial Natural Science Foundation of China under Grant BK20150637, by Ant Financial Research Fund. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. Xu, G., Yu, W., Griffith, D., Golmie, N., Moulema, P.: Toward integrating distributed energy resources and storage devices in smart grid. IEEE Internet Things J. **4**, 192–204 (2017)
2. Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A survey on internet of things: architecture, enabling technologies, security and privacy, and applications. IEEE Internet Things J., Enabling Technologies (2017)
3. Altium Limited: Philips RC5 Infrared Transmission Protocol, 13 Sept 2017. http://techdocs.altium.com/display/FPGA/Philips+RC5+Infrared+Transmission+Protocol
4. Altium Limited: NEC Infrared Transmission Protocol, 13 Sept 2017. http://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol
5. Mosquitto (2018). https://mosquitto.org/
6. Adafruit Industries: TVBGone, 4 Jan 2018. https://cdn-learn.adafruit.com/downloads/pdf/tv-b-gone-kit.pdf
7. Gotschlich, M.: Remote Controls – Radio Frequency or Infrared White Paper, Infineon Technologies AG (2010). https://www.infineon.com/dgdl/RF2ir+WhitePaper+V1.0.pdf?fileId=db3a30432b57a660012b5c16272c2e81
8. Raspberry Pi Zero W (2018). https://www.raspberrypi.org/products/raspberry-pi-zero-w/
9. Christoph Bartelmus: Linux Infrared remote control (LIRC), 26 May 2016. http://www.lirc.org/
10. Schwind, B.: Sending Infrared Commands From a Raspberry Pi Without LIRC, 29 May 2016. http://blog.bschwind.com/2016/05/29/sending-infrared-commands-from-a-raspberry-pi-without-lirc/
11. Sensacell: IR Rememberizer- IR Remote Control Recorder/Player, 6 Aug 2014. https://forum.allaboutcircuits.com/blog/ir-rememberizer-ir-remote-control-recorder-player.648/