ARINA:
Arduino Remote Infrared Network Adapter

René Neff, Thomas Trimborn and Matthias Wübbeling

Institute of Computer Science 4, IT Security, Universität Bonn, Germany
{neff, trimborn, wueb}@cs.uni-bonn.de

Abstract

In this paper we present ARINA, a light-weight Arduino based system for forwarding infrared signals via Ethernet. Short introductions on infrared communication and Arduino as a development platform will be given. A transport protocol will be defined and its usage explained throughout the projects software design. Full functionality of all built hard- and software components will be shown in the evaluation which will be supported by conducted tests. Finally, we will present possible future extensions and their potential use in smart home environments.

1 Introduction

This project aims to create a compact solution to operate infrared controlled devices without the need of direct line of sight. It even provides full control over multiple stationary infrared controlled multimedia devices via a mobile Ethernet based network adapter. Having a TV set-top box hidden away or start recording channels at home from every location around the globe are some of many possible use cases one might think of. Although many modern devices already support different kinds of control, there is no universal solution that supports multiple endpoints. Unlike such an app, there is one common basis: nearly all multimedia devices still use infrared remote controls. To provide a non-line of sight control we develop and implement a way to receive, encapsulate and display infrared signals, as shown in figure 1. We aim to create an economical and easily recreatable solution. First, we investigate related work and provide a basic set of background information regarding infrared signaling and the chosen Arduino platform [9]. We follow up with an overview of the software design, including a self-defined protocol for encapsulating infrared signals for the purpose of transportation via a TCP network connection [8]. The implementation is presented including a full list of components and the corresponding hardware connection layout. Finally, we present an evaluation of the built hard- and software components and the results of the tests with different kinds of multimedia devices, ranging from television to toy helicopters. A short outlook on future work will concludes this paper.

Figure 1: A typical ARINA setup
2 Related work

Different solutions for slightly similar purposes are already commercially available. Products from Logitechs Harmony series [3], Philips Prestigo [5] or OFAs URC8800 [4] are starting from prices about 70 €. All those products feature some kind of single hub which interacts with multimedia devices via infrared signals and an app used on a smartphone or tablet. Those devices rely on pre-defined protocols fitted to their product specific infrared signals. At best these solutions allow users to have unknown commands added into their control options. As far as our research conducted, none of those products support an open protocol for interaction and therefore no extension beyond their predefined purpose. Special purpose products by IRTrans GmbH are based on an ATMega32/64 microcontroller, they use closed source hard- and software to receive, forward and transmit infrared signals [1]. As an easily available and well known platform to tinkering hobbyist we base our work on an Arduino Mega 2560. The selected Arduino provides a single-board microcontroller based on an 8-bit Atmel AVR. It features an USB interface, 16 analog input pins, as well as 54 digital Input/Output pins which allow to attach various extension boards [2]. Our setup is powered by a connected battery which lasts for about 20 hours or a connected USB power source for longer use. With ARINA and the self-defined transmission protocol, we open up many different options which exceed just turning on TV or switching to DVD playback. ARINAs open protocol and its support for unknown infrared protocols allow many different ways of interaction, even automation become possible, please refer our outlook on future work in chapter 7.

Figure 2: Electromagnetic spectrum
3 Research on infrared

To send and receive data over infrared it is necessary to take a closer look on what infrared light is and how to send and receive data. As we know how to transfer data over infrared connections, we determine how data is encoded in infrared remote controls and the meaning of each transmitted bit. Infrared light is electromagnetic radiation in the spectral range between visible light and microwave radiation as shown in figure 2. Its wavelength ranges from $10^{-3}\text{mm}$ (1mm) to $7.8 \times 10^{-7}\text{mm}$ (780nm) and is not visible to the human eye but easy to produce and measure.

3.1 Infrared signaling

Infrared remote controls emit modulated signals with a frequency around 36kHz and 38kHz, some manufacturer also use higher or lower modulation frequencies [6]. A short impulse of infrared light is called a burst. Lining up a sequence of bursts at a given carrier frequency which are interrupted by breaks gives one half of the data used in command sequences of remote controls. The second half is described by the pause, where no light is emitted. Out of these two blocks all signals composite a unique signal. A block of bursts is interpreted as a logical 1 and a block of pause is interpreted as logical 0. The specific command is defined by the specific protocol used and is not represented directly by the actually received data. The interpretation depends on the specific manufacturer’s application protocol, as described in section 3.2. Infrared diodes are light emitting diodes (LED), emitting light in the infrared spectrum, if electric power is sent in forward current of the diode. Infrared diodes are produced with support of a spectral range between 840nm and 950nm and most infrared remote controls use diodes with around 940nm. These contain aluminum gallium arsenide which is the cheapest material in capacity emit infrared light. There are some issues when the infrared spectrum is concerned, as it is impossible to detect a single source for recognition, because everything emitting energy (such as a body’s temperature) produces infrared light. To filter the specific infrared signal from background noise, the signals are represented by short bursts of light with a specific frequency set by the used protocol. The receiver ignores signals of different frequencies. The recognized phases of transmission and pause do not necessarily correspond with former mentioned 1s and 0s. Therefore, it is needed to build the bitvector out of a combination of bursts and pause in a fashion that two consecutive set bits are safely detected and transmitted and not misinterpreted by the receiver if anything crosses the line of sight between sender and receiver. To send modulated infrared signals via an electric circuit, pulse-width modulation (PWM) outlets are used on the sending ARINA. This allows emission of short flashes of light at the exact frequency expected by the receiver. Infrared radiation is absorbed and transformed into electric currency by photodiodes. The resistance in back direction of the photo diode

![Figure 3: As modulated infrared radiation transmitted data is transformed back into a digital signal](http://sbprojects.net/knowledge/ir/index.php)
lowers, if light in the specified wavelength is absorbed. The received signal is reinforced, limited and sent through a band-pass filter to suppress background noise before forwarding it to a demodulation unit. This, as presented in figure 3, releases a digital signal which does not contain modulated bursts but electricity switched on and off accordingly. All these functional parts are integrated into the integrated circuit (IC) thus, an assembler of an ARINA device can easily use it. This range can be covered by one single IC, even if it only supports a subrange, if it is possible to fill the gaps occurring when receiving a signal of another frequency, as shown in figure 4. Additionally, it is possible to use more than one receiver for different frequencies, which is not yet supported by the ARINA software.

Figure 4: shift of timeslots in 0,5ms at given frequencies

3.2 Known and unknown protocols of infrared transmission

Infrared remote controls are shipped with several multimedia devices, often incompatible to each other. This leads to living room tables collecting a lot of different remote controls, which would not interact with devices of other brands. There are software libraries to control all devices with one microcontroller or a personal computer which acts like a universal remote control. Such libraries implement different protocols of different vendors. They contain descriptions of a preamble and how it is used, as well as how a command is represented, i.e. the order of sent data of the supported devices. A library can be saved and used on every ARINA, but it can always happen that an unknown protocol is used by a multimedia device, in which case it is impossible to interpret the received data. In that cases, data has to be transported from and restored to in its exact representation. To recognize the form of the data sent, all devices have to know about the protocol used. Lots of different protocols are implemented in devices nowadays. Two of the most frequently used are described later on to give an overview of differences between them.

The NEC protocol is commonly used in Japan and other Asian countries but also appears often in Europe and America. It transmits an 8 bit address and an 8 bit command in each packet and both are sent twice to minimize the detection failure rate. The infrared signals are transmitted on a 38kHz modulated signal as shown in figure 5, containing the representation of 1s and 0s. As shown in figure 6, each message starts with a burst of 9ms followed by 4.5ms pause. This defines the preamble of the protocol, followed by the address twice, with the second one having all bits inverted. After that the command is also sent twice, with inverted bits on the second transmission. Due to the second transmission of all bits, every message lasts 67,42ms. If a button on the remote control is pressed longer than the transmission of the message, the same message is sent repeatedly every 110ms which consists of a 9ms burst, a 4,5ms break and a 560µs pause.
Philips RC-5 and RC-6 are also widely deployed and RC-6 is based on RC-5, extending it e.g. providing a variable command length. RC-5 contains a 5 bit addresses and a 6 bit command modulated on a frequency of 36kHz and a constant bit duration of 1,778ms, as detailed in figure 7, showing 36 units of the 36kHz signal. First two 1s are transmitted, thus, a message always begins with a pause of 889µs, followed by a switch bit. This bit is inverted every time the button on the remote control is released or pressed again. Following this preamble, the address and the command is sent, as shown in figure 8. If the button is continually pressed, the message is repeated every 116ms.
4 Software Design

To support future use and development beyond this specific project, ease of adoption and modularity where major design goals. This benefits beginners and experienced developers likewise as it provides an easy point to get started with a project using single components of ARINA. Additionally, we used Doxygen for documenting the project, providing a consistent documentation. One has to address different limitations when developing microcontroller software, those range from limited computing power to limited memory up to limited space to save program code or data in general. This led to a focus on using the call by reference principle throughout our code. With use of an external infrared signal decoding and encoding library [7] we covered the most common infrared signals encodings. To circumvent the limitaion of implemented encodings we had to cover the case of a raw encoding transmission, which is used as a fallback in case of failing identification of the used infrared signal encoding. As our previous conducted research revealed, there is no standardized way of transporting decoded or raw infrared signals via Ethernet. Therefore, we developed and defined a new protocol which serves this purpose explicitly.

4.1 Protocol

Ethernet and IP are used in conjunction with various protocols which utilize their standardized features, so do Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). A single UDP package can hold up to 65535 bytes, which suffice to transport an encoded infrared signal. Each received infrared signal should be send as fast as possible to be available at the remotely stationed ARINA, this results in a steady stream of data packages. As the order of sequential infrared signals is important, order preserving mechanisms are necessary. Besides order, lost packages, therefore lost commands, are also to be avoided. Additionally, there are media devices responding to given commands by sending infrared signals themselves. Finally, data transmission should not suffer from more loss than necessary, which ultimately led us to choose a reliable, ordered and error-checked protocol and therefore TCP over UDP as transmission protocol.
The Arduino Ethernet Shield and its corresponding Ethernet library, segments provided data into single byte-sized fragments. Thus, we chose to focus on this entity size when designing our custom protocol. To be able to even support future multimedia devices and their possibly new way of encoding each single commands (e.g. ‘Play’ or ‘Stop’) in infrared signals, we used well defined infrared signal encoding protocols as for example the widely used NEC protocol, refer protocol section 3.2. Those encode long infrared signal sequences in a reproducible way so they can be represented by, in comparison, shorter long integer values. Additionally, this saves the need for a large code to command database for every possible, already available multimedia device and those to come. We see this as a fair tradeoff between recognition of a single command and the size of data to be sent. To be able to transmit infrared signals in the way previously described, we designed the software to recognize whether we were able to identify the received infrared signal. If so we are thereby able to decode it by matching it with known infrared signal protocols, therefore we have to transmit a protocol identifier and the corresponding decoded value. Otherwise, if decoding fails, ARINA falls back to raw data transmission. This leaves us with two different cases in which our protocol has to transmit data, as shown in figure 9. The designed protocol is defined upon a byte vector which includes a header section followed by a payload section. The header section holds the infrared protocol used to decode the data within the payload section, which is a 32 bit long integer value. Additionally, a header field contains the length of the following payload section. Payload will then be the encoded infrared signal closed by a byte, which provides the number of set bits of the reconstructed long integer value. In case of raw data transmission, as illustrated in figure 10 we define the protocol header flag ID to be ”-1” in two’s complement (Binary: 11111111). The payload section then contains single infrared pulses coded into binary representation by describing the duration of every burst and pause in 16bit for each time slot.

Figure 9: Flowchart: Transmission of IR signals via ARINA
5 Hardware

The ARINA is based on an Arduino Mega 2560 [2] with an Arduino Ethernet Shield and several other components described later in 5.1. The Arduino itself supports 14 PWM outlets, utilized to send an infrared signal modulated with a defined frequency as described in 3.1. The Arduino Ethernet Shield contains a 100Mb/s ethernet port as well as a micro sd cardreader. The Ethernet connection is used to connect to another ARINA component and the SD cardreader is needed to read configuration files, to support a comfortable configuration of the devices. An example device is shown in figure [11].

Figure 11: Arduino Mega 2560 with Arduino Ethernet Shield

Picture from onemansanthology.com
5.1 Components

In addition to the Arduino, a infrared diode by Harvatek model descriptor HE3-290AC is connected. It is built in 5mm structure and it emits infrared radiation at a wavelength of 940nm. The infrared receiver module used is a OS-1638 by OS-OPTO, which is capable to detect infrared signals with a wavelength about 940nm and a frequency of 38kHz. Both parts will work at 5V current, which is provided by a stabilized output port of the Arduino. Additionally, we use three LED in different colors to represent the different states of the software. These parts are assembled on a breadboard, cables and resistors to the Arduino Mega 2560 are wired as shown on the circuit diagram in figure 12.

Figure 12: Circuit diagram of an ARINA unit

5.2 Handling

To use the described ARINA solution, one has to setup the configuration files for each device and to ensure that they are connected via Ethernet. If powered on, both ARINA will indicate the initial state with the red LED on each breadboard. After pressing one of the buttons of a infrared remote control the blue LED flashes on both devices, signaling that data is being transmitted. The red LED turns off and the green one on, as soon as a connection between both devices is established.
6 Evaluation

The task of the project group was to build a system to transfer infrared data, received on a distant location, through existing networks and replay the command at another site. This evaluation shows the capability of ARINA to do so (6.3). Furthermore, functionality of receiving, transmitting and sending loops (6.1 and 6.2) with ARINA are shown. This section also explains the structure of experimental assemblies and discusses the results of related experiments.

6.1 Direct reflection

This experiment is designed proving the concept of receiving and sending infrared data in general. Software is needed, receiving an infrared message end replying it five seconds later on the same ARINA device which received the original message. The functionality is evaluated on two different devices of ARINA. The first multimedia device is an Epson projector (type EB-1771w) which uses the well known NEC protocol (see section 3.2) and the second one is a Samsung Television unknown model using an unknown protocol. After pressing randomly selected buttons on the related remote control the device responds immediately on the signal and five seconds later on the repeated signal sent by our ARINA.

![Diagram](image)

Figure 13: Field-Study setup
6.2 Reflection loop

After successfully conducting the first experiment, a second ARINA device is built up and software to receive, transmit, and send infrared signals is used. To test the interconnection, a loop is created, where an infrared signal is initialized by pressing a button on a remote control aimed at one ARINA, which receives the signal, transmits it to a second ARINA over Ethernet where it is emitted and sent back again to the first ARINA via infrared signals. During our tests each signal was recognized every time by the device shipped with the used remote control, to check if it is still solid enough to transmit the desired control message. On average a signal was recognized for about four loops up to fifteen loops in some cases. The quality of a particular transmission always was close to the original signal, with little variance at the length of burst and pause sections, which is caused by the low sampling rate of Arduino input pins.

6.3 Final evaluation

The final evaluation setup was built up in several places to test the setup with as much different devices as possible to assure the functionality in most cases. The network connection was established by direct link, via a switched Ethernet. In some cases we used the internal network of the Institute of Computer Science 4 of the University of Bonn, connected over a VPN link from outside the universities network. The general structure of the evaluation setup is shown in figure 13. To evaluate the full functionality, each and every key is pressed on the original remote control belonging to the tested device, checking the specific device at another location for the expected result. All results are outlined in figure 14.

<table>
<thead>
<tr>
<th>Device name</th>
<th>Type</th>
<th>Producer</th>
<th>Protocol</th>
<th>All keys</th>
<th>Purchased in</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB-1771W</td>
<td>Projector</td>
<td>Epson</td>
<td>NEC</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>UE40EH5000</td>
<td>TV</td>
<td>Samsung</td>
<td>RAW</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>BD-E5500</td>
<td>3D-BR-Player</td>
<td>Samsung</td>
<td>RAW</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>42LV579S</td>
<td>TV</td>
<td>LG</td>
<td>NEC</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>DVX592H</td>
<td>DVD-Player</td>
<td>LG</td>
<td>NEC</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>Connect ID 32</td>
<td>TV</td>
<td>Loewe</td>
<td>RC-5</td>
<td>yes</td>
<td>2009</td>
</tr>
<tr>
<td>230E</td>
<td>DVD Player</td>
<td>Redstar</td>
<td>NEC</td>
<td>yes</td>
<td>2001</td>
</tr>
<tr>
<td>55FU4243</td>
<td>TV</td>
<td>Thomsen</td>
<td>NEC</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>Firestorm IR</td>
<td>Mini-Helikopter</td>
<td>AMEWI</td>
<td>RAW</td>
<td>yes</td>
<td>2012</td>
</tr>
<tr>
<td>SoundTouch</td>
<td>HiFi</td>
<td>Bose</td>
<td>RAW</td>
<td>yes</td>
<td>2014</td>
</tr>
</tbody>
</table>

Figure 14: Table of field-study observations

There column protocol means, whether the used protocol was recognized or if raw data was transmitted, 'all keys' indicates that all submitted keys worked as expected after transmitted by ARINA. These values show that the ARINA system works fine in different network environments and with a lot of different devices.
7 Future Work

As stated earlier, many different use cases could be realized using the presented ARINA approach and its specially designed protocol. A system extended by a central server could support automated events which might depend on different user defined triggers. It is easily possible to use it as an automated recording system even with older devices not having a built-in Personal Video Recorder. In combination with an electronic program guide it may be possible to also record shows according to predefined keywords included in a shows title or description. One might also think of a simple intruder scare system, turning on and off a TV or switches through different channels, thereby creating an illusion of an individual being at home.

Arduino supports multiple extension shields. Thus, different interactions via bluetooth, Wi-Fi or other ISM band based could be added to the already available infrared connection. The already used ethernet shield also supports power over ethernet with a small add-on, which would render any other connected power resource obsolete.

References


All webpages last visited on 16.04.2014.