Pico-Remote-Analyzer

Firmware Version 1.00
User Guide
Updated January 18th, 2023

IMPORTANT :
This User Guide is about Pico-Remote-Analyzer Firmware Version 1.00 from Andre St. Louys.

Join our Pico-Remote-Analyzer discussion group on:
https://github.com/astlouys/Pico-Remote-Analyzer/discussions

Index

Pico Remote Analyzer User Guide

Do you want to share your experience with Pico-Remote-Analyzer and help other users ?
Join our discussion group on:
https://github.com/astlouys/Pico-Remote-Analyzer/discussions

Among the subjects of interest:

What features are missing?
Did you find some bugs (what are they)?
Did you improve the Firmware (how)?
Did you add some more features (which)?

Let us know!

If you want to send me a personal email (as long as it is something constructive),
here is my email address:

Andre St. Louys
(Quebec, Canada)
astlouys@gmail.com

About Pico-Remote-Analyzer Version 1.00

Version 1.00 is the first release of this Firmware.

Introduction

Did you ever want to add a remote control to one of your Raspberry Pi Pico projects? Have you lost one of your remote control and thought about replacing it with a "Pico-Remote-Translator", translating another remote control that you have to emulate the one you lost instead of buying a new unit? Have you thought about adding "macro commands" to one of your remote controls?

If you answered "yes" to one or more of the questions above, this Firmware may help you reach your goal…

Basically, Pico-Remote-Analyzer will help you understand the underlying timings and allow you to decode one of your infrared remote controls, so that you could recognize the commands sent by the remote control and add functionalities to your project.

You may want to take a closer look at the Pico-Green-Clock (in my repository) if you want an example on how to add a remote control to a Pico project.

As mentioned on the cover page, this User Guide is about Firmware Version 1.00 from Andre St. Louys and it is the first release of this utility.

It is based on a Raspberry Pi Pico and only two external parts / devices are required:

1) One VS1838b infrared sensor.
2) One active buzzer ("active" means that there is an oscillator already integrated in the buzzer, so that it will sound as soon as a 3.3 volts source is applied to it).

This User Guide assumes that you are already familiar with basic electronics and basic C-Language programming with the Pico. For those who are not, you may want to find information on the Internet and come back to the Pico-Remote-Analyzer later.

Understanding remote control analysis

I've seen other projects about remote control with discussions on "Address part", "Command part" and "Checksum part" of the infrared burst from the remote control. It may be interesting from a user perspective to find more information about the specific remote control unit under analysis to understand all the details of the infrared data transmitted by the remote control. However, the goal of the current utility is simply to "recognize" the command sent by the remote control, in such a way that if the same command is sent again later, the Firmware will be able to perform some specific action. Basically, an infrared burst will translate to a hexadecimal value. Even if we don't know "what are the different parts" of this hexadecimal value ("address", "command", "checksum", and so forth) the fact that we can recognize it means that we can take a specific action when the command is sent again and decoded.

I understand that a checksum could be used to validate the data stream received. However, the command decoded will be compared to the list of the valid commands that we analyzed with the Pico-Remote-Analyzer. If the infrared burst received from the remote control is found in the list, the function associated with this command will be executed. If the command is not recognized, the infrared burst will simply be ignored.

Most remote controls send data on a 38 kHz carrier. Fortunately, the VS1838b used in this project will take care of this carrier and will convert the infrared burst to a series of logic levels (0's and 1's) that Pico-Remote-Analyzer will translate to a hexadecimal value.

Again, for those who need to be convinced, you can take a look at the Pico-Green-Clock. Such a remote control has been added and it's been working without a glitch for the past 12 months…

Pico-Remote-Analyzer is interrupt-driven. When the infrared sensor changes its logic level, an interrupt is triggered on the Pico and current timer values will be memorized so that the length of every "Low level" and every "High level" is sampled. By analyzing those timings, we can give a sense to the data stream received.

Two-steps analysis

IMPORTANT: Before going further, it is important to understand (at a high level) how the remote control analysis is done with the Pico-Remote-Analyzer, called a "2-steps analysis".

Step1: The first step is to make a raw timing analysis of the infrared bursts sent by the remote control. You will need to analyze the stream sent by 7 or 8 remote control buttons to get the timings typical to your remote control unit. There is a lot of data, but it will be sent to a log file and can then be analyzed with your text editor. We will follow an example to see how to analyze those data streams and what information is important to extract from them.

Step 2: Once you have analyzed 7 or 8 examples of the timings specific to your remote control unit, you will be able to adapt the decoding function / algorithm so that it matches the specific timings of your remote control. You will need to modify the C language decoding function and rebuild the Firmware required to go through Step 2. Hopefully, this may be only a few #define`s to be changed (or maybe not, depending on the remote control unit). This may sound complicated, but if you go step by step, examples in this User Guide should help you achieve your goal with success.

Operation sequence

The sequence of operations is important if you want to go up-to-speed in a quick time. Here is a summary of the sequence you must follow. You will find the details about each operation below in this User Guide.

1) Identify the remote control unit to be analyzed.
2) Install the infrared sensor (VS1838b).
3) Install the active buzzer.
4) Configure a PC computer as a terminal emulator.
5) Download the software to the Pico and run it.
6) Start the terminal emulator program.
7) Enter the brand of the remote control to be analyzed.
8) Enter the model number of the remote control.
9) Press a button on the remote control.
10) Select "Display burst timing" in the Firmware menu.
11) Repeat steps 9 and 10 seven or eight times.
12) Analyze the timings of the remote control.
13) Adapt the Firmware decoding algorithm.
14) Decode all remote control buttons.
15) Display all remote control buttons and commands.
16) Integration of a remote control in a program.

Refer to the numbers above in the list to find the corresponding section in this User Guide.

1) Identify the remote control unit to be analyzed

There is not much to say about selecting a remote control unit to be analyzed. However, it is important that the unit be an infrared unit. There are some remote control units that are based on radio frequency ("RF") instead of infrared. Typically, an infrared remote control has a dark red plastic in front of the unit. Also, when used with you usual device (television, sound system or whatever), if you hide the front of the remote control with your hand, it will stop working, whereas the RF remote control will continue to work without problem.

You may want to select a well know brand for your first try, since it is most probably using a more standard framing and will be easier to understand / analyze as a first exercise.

2) Install the infrared sensor (VS1838b)

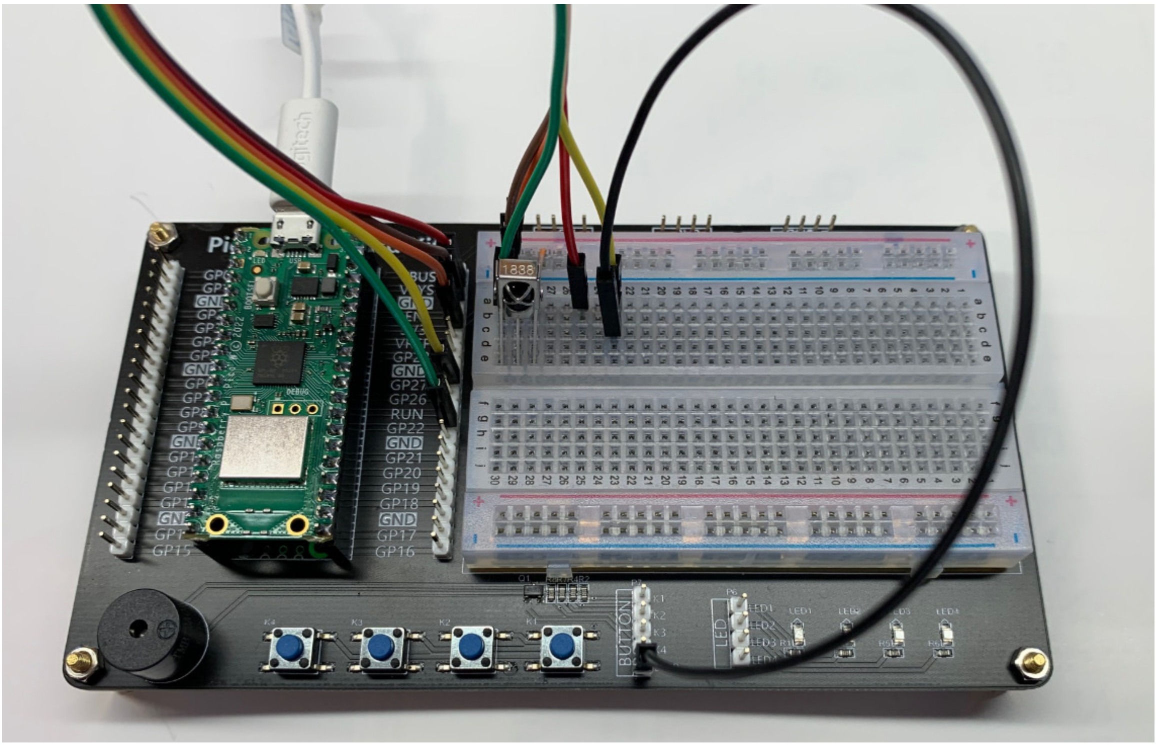
The first step is to install an infrared receiver / sensor. The VS1838b IR receiver is not expensive and it takes care of filtering out the 38 kHz carrier of most remote controls.

VS1838b infrared receiver

I used GPIO22 to receive data from the VS1838b IR sensor on the Pico, but any GPIO may be used, as long as you change it in the source code, rebuild the code and re-flash it to the Pico.

Be aware that even if the infrared sensor can work with 5 volts, the Pico must receive 3.3 volts logic levels. So, you must feed the VS1838b with 3.3 volts.

I used the “Pico Breadboard Kit” from Geeek as the platform for the Pico-Remote-Analyzer (see picture below). I also used a Pico W that was on hand, but no wireless options are actually used, so a “plain Pico” will do the job as well.



Pico Breadboard Kit from Geeek

You can see the VS1838b in the middle of the picture (at the left of the protoboard), and also the active buzzer (discussed in the next section) that is already provided with the Pico Breadbord Kit (the round black cylindrical part at the complete bottom left).

3) Install the active buzzer

Even if currently not used extensively in the Pico-Remote-Analyzer project, the active buzzer makes things easier as we go and it may also be used to cover more functions as later versions of this Firmware are released. Such a buzzer has 2 terminals: one for ground and another one to receive the 3.3 volts that will turn it On.

I install the active buzzer on GPIO 27 in the Firmware, but once again, any GPIO may be used, as long as you change it in the source code, rebuild the code and re-flash it to the Pico.

As mentioned previously, "active" buzzer means that there is an oscillator already integrated in the buzzer, so that it will sound as soon as a 3.3 volts source is applied to it.

Be aware that "active" and "passive" buzzers are physically very similar. However, whereas the active buzzer will sound as soon as you apply a ground and 3.3 volts to it, the passive buzzer doesn't have an integrated oscillator and must be driven by a source or varying frequency corresponding to the sound to be heard. If you want to make sure your buzzer is an "active" type, use one of the two methods below:

1. Connect the buzzer directly to a 3.3 volts source. An "active" buzzer will make a sound whereas a "passive" buzzer will only make a quick "click". (Note: You can also use 5 volts to directly feed the active buzzer for this test, but remember that the Pico logic levels must be 3.3 volts).

2. Check the buzzer with an ohmmeter. An active buzzer will typically have a resistance of several hundred ohms whereas a passive buzzer will have a low resistance value around 8 or 16 ohms.

4) Configure a PC computer as a terminal emulator

Introduction

The Pico-Remote-Analyzer will display a lot of information for analysis. Even if a small LCD display could be used, it won't be practical for all data being manipulated. Also, using a PC with a terminal emulator program will allow saving all Pico-Remote-Analyzer working sessions to a log file for further analysis / reference.

This section will show you how to make a connection with your PC to display Pico-Remote-Analyzer information. As mentioned previously, working with an external terminal emulator (80 X 24 characters on the screen or more) is much easier to work with, and also much faster to read than looking at a small LCD display with only a few lines of data at a time.

Pico to computer, USB-to-USB

The easiest way to communicate between the Pico and your computer is probably with a USB-to-USB connection. Make sure your Pico-Remote-Analyzer make file (CMakeLists.txt) contains the following lines:

Pico_enable_stdio_uart(Pico-Remote-Analyzer 0)
Pico_enable_stdio_usb(Pico-Remote-Analyzer 1)

The first line will stop sending display information to the Pico's UART and the second line will make it sent through the Pico's USB port.

Then find a terminal emulator program that will work on your computer. I use the popular TeraTerm, Freeware and Open source on Windows. As its author wrote, "it is not a full-fledge terminal emulator". However, I found it to be the perfect tool to display information sent by the Raspberry Pi Pico. If properly setup, it will also save all received data to a log file for further analysis, which is important in our case.

https://download.cnet.com/Tera-Term/3000-2094_4-75766675.html

This is it for now. We will return to TeraTerm in section 6 below.

5) Download the software to the Pico and run it

The "Pico-Remote-Analyzer.uf2" file is already available from my GitHub repository. Flash it to the Pico. (Note: It is assumed that you already know how to proceed to do so).

You may want to make modifications to the Firmware for some reason, but using the original "Pico-Remote-Analyzer.uf2" from my GitHub repository for now will make sure that the rest of the setup works fine for your first try (that is, the Pico with the VS1838b and active buzzer, terminal emulation program, etc…)

If you properly followed all previous instructions and if everything is properly configured, you should now hear the active buzzer making a quick "beep" every 2 or 3 seconds. This is to remind you to start the terminal emulator program at this point (see next section).

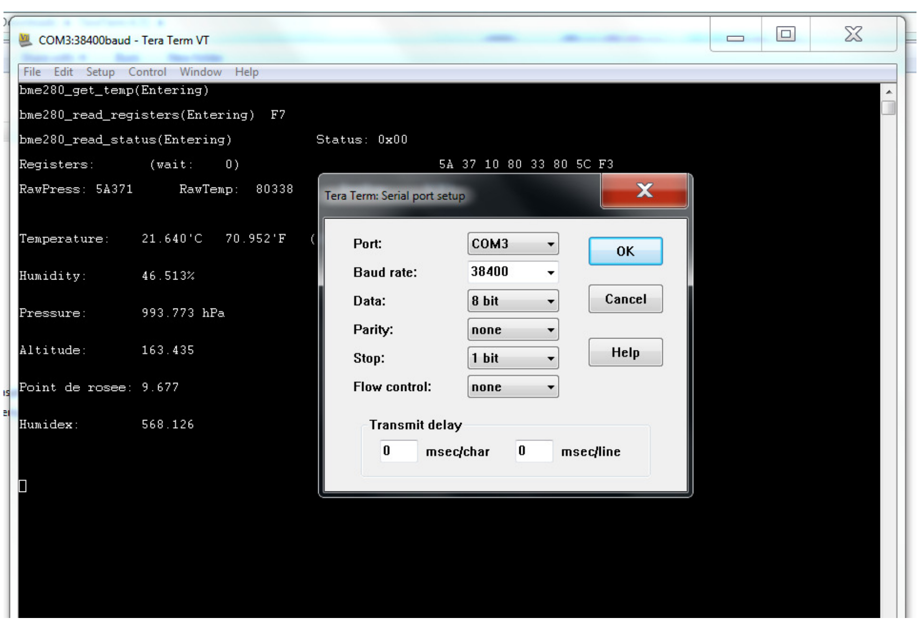
6) Start the terminal emulator program

Back to your PC, the instructions below assume that you are using TeraTerm, but you can use another program if you want. Now, proceed as follow:

1) Start the terminal emulator and go to the "File / New connection" menu, you will see that there is a "Serial" option and a serial port number should have been assigned to the serial-to-USB adaptor. Note: on some older Windows versions, you may need to install a special USB CDC driver for the adaptor to be recognized as a COMx (serial port) device. (Search information on the Internet if this applies to your case).

2) Then, take a look at "Setup / Serial port" menu. A default serial port may be proposed as a default if the USB CDC has been recognized by the system. Also, the serial protocol must match the one of the Pico. (921600, N-8-1 and no flow control if you didn't change this setting from the original source code).

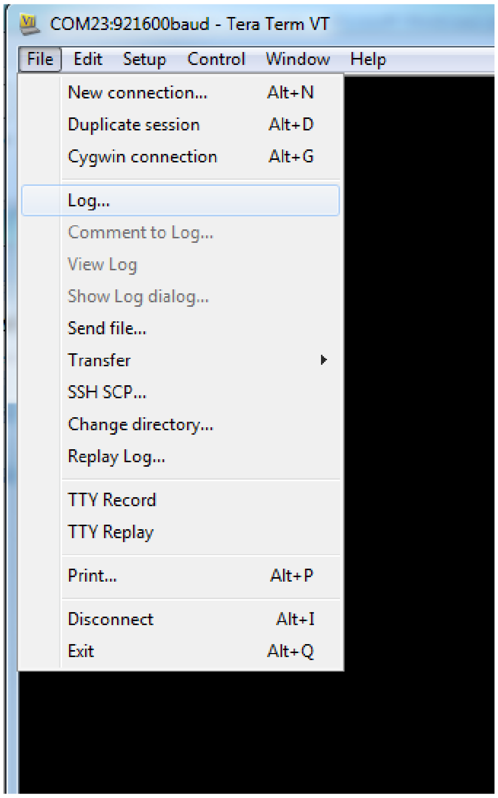
NOTE: TeraTerm will give an error if you try to connect the PC to the Pico's USB port when the Pico-Remote-Analyzer Firmware is not started. On the other hand, when the Pico-Remote-Analyzer Firmware is started, it may be already too late to see / log the first information displayed on the screen. For this reason, a loop has been added at the beginning of the Firmware that beeps (with the help of the active buzzer) until the USB CDC communication has been established. While the system waits in the loop, user has the time to start the terminal emulator program. Once the Pico-Remote-Analyzer is connected, the terminal program automatically senses the USB connection and go on with the Firmware as the Pico gets out of the waiting. Take a look at the source code for more information.



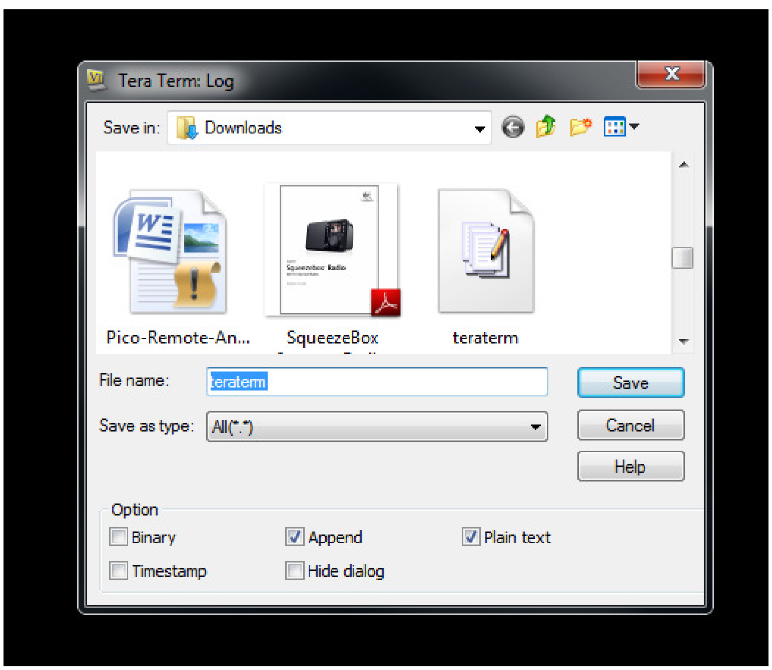
TeraTerm: Serial port (USB CDC) setup

Pico Remote Analyzer User Guide

Once TeraTerm is started and the serial port properly setup, start logging to the log file. You can access log file setup under the "File" on the menu bar (see picture below)



TeraTerm: Access to the log file setup



TeraTerm: Assign a name to the log file

7) Enter the brand of the remote control to be analyzed

When the Firmware is first started, a remote control brand name is proposed by default (we will see later where it comes from). You must enter / overwrite the proposed name with the brand of the remote control that you will analyze. This name will be displayed in the log file so that if / when you refer to those log files later, this important information will be available to you.

8) Enter the model number of the remote control

Similarly to the previous section, you must now enter the model number of the remote control unit that you will analyze. The same company (“brand”) usually manufactures many different remote controls for its different products (TV, sound system, CD player, BluRay disks, etc…). It is a good idea to keep the model number of the remote control unit (not the product that it controls) so that it is also saved to the log file (I personally also add a picture of the remote control along with the remote timing information).

The Firmware now indicates “TBD” (To Be Determined) as the remote control model number. Enter the model number (usually indicated on the remote control unit itself) or at least, an indication that will help you to identify the unit later if required.

9) Press a button on the remote control

At this point, you can see on the screen that the Current step count is 0. The “step count” represents the number of “logic level changes” sent by the remote control unit when a button is pressed (also called an infrared “data stream”). When the number of steps is zero, it means that Firmware variables are “initialized to zero” in the Pico-Remote-Analyzer and ready to receive an infrared data burst from the remote control.

Point the remote control toward the little receiving window of the infrared sensor (VS1838b) and press a button. As soon as the Firmware receives the infrared data stream, it should display a header, the total number of steps received and a menu. If not, something goes wrong and you must find where the problem comes from and fix it. If everything is fine, proceed to next step.

10) Select “Display burst timing” in Firmware menu

If you remember what has been said in the section “Understanding remote control analysis” previously in this User Guide, the analysis of the remote control is a 2-steps procedure. The first step translates in the Firmware by the “Display infrared burst timing” menu choice. This is the selection that you must do right now.

When doing so, the Firmware will ask you to enter the identification of the remote button that you pressed. Like for the brand and the model number, this information will be kept in the log file so that you can refer to it in the future.

A table of timings similar to those shown in Appendix A should appear on the PC screen

NOTE: Obviously, the total number of steps and / or the values associated with all timings could be different since you most probably don’t use the same remote control brand and model number than the one used to generate the timing tables of Appendix A. Nonetheless, you should see a similar table). We will examine this information in details in section 12 below. For now, go on with next step.

11) Repeat steps 9 and 10 seven or eight times

At this point, the Firmware should ask you to press a remote control button, so you are ready to proceed with step 9 again.

Make sure the PC displays a timing table similar to those in Appendix A and repeat steps 9 and 10 above to display the timings of 7 or 8 remote control buttons.

NOTE: It is not always the case, but usually, all data streams from the same remote control will have the same number of steps. So if you see that the step count is consistent from one button to the other, it is a good indication that everything goes fine. If step count is different, you may want to make a retry with the same remote button. We will see later why you can sometimes get a step number different.

12) Analyze the timings of the remote control

The analysis below will be done with the help of the timing tables shown in Appendix A. As previously mentioned, chances are that some details of your timings will be different. However, the “thinking” that follows should help you do the equivalent with your own timing tables and guide you about how to proceed with your own remote control.

Following are observations that can be done while taking a closer look at the timing tables in Appendix A. Some observations are based on the format of the documents, whereas others go deeper in some details. They are given in no special order, except maybe that the first observations are the most obvious, going to less obvious ones as we go deeper in the analysis. Even is most analysis apply to all timing tables, when specific values are given, they are taken from the first timing table (button “Power”).

Observations

- The header gives general information, like the type of microcontroller used (Pico or Pico W), the Pico’s Unique ID (similar to a serial number), brand name and model number of the remote control being analyzed and the step count of the infrared burst that follows, that is, the total number of logic level changes sent by the remote control unit (translated as 0’s and 1’ by Pico-Remote-Analyzer.

- Under the header, we see the remote control button used to generate the timing information that follows.

- The number of lines in the chart corresponds to the step count plus one (136 instead of 135), but we see that step 136 is 0 and its logic level is “---“, which means that it is undefined (not a “0” and not a “1”). So, the last line is there only as a proof that we are done will all pertinent information in the previous lines.

- Each page is divided in two sections: the right columns are the continuation of the left columns.

- Each “button” on this remote control unit gives 2 pages of information (we see that the second page begins with step 100, which is the continuation of the first page).

- Beside each step number, we see what logic level corresponds to the entry: Low for 0 volts and High for 3.3 volts.

- Beside the logic level, we see the duration of this step in microseconds.

- Since the GPIO line where is connected the infrared sensor (VS1838b) is hold with an internal pull-up resistor while the line is idle, not surprisingly, the first step represents the first change of the logic level and represents a "Low" logic level, as indicated beside step number 1.

- The infrared stream for all buttons is made of 135 steps. (step count is the same for all buttons analyzed).

- Almost all step durations have different values. However, most of them have values around either 550 microseconds or 1675 microseconds. Considering normal imprecision from the remote control and / or from Pico-Remote-Analyzer behavior, we can "standardize" values to 550 and 1675 usec.

- There are two exceptions about this "550 or 1675" microseconds for the first two steps which are around 4450 microseconds.

- There are three other exceptions on this "550 or 1675" microseconds for the steps 68, 69 and 70.

- All "Low level" steps have a duration of 550 microseconds (the only two exceptions being steps 1 and 69).

- All steps with a duration of 1675 microseconds are High levels.

Interpretations

What are the deductions / interpretations that can be done, given the observations noted above?

After reading some documentation and making the analysis for different remote unit brands and model numbers, here are what I ultimately come up with:

(NOTE: When specific values are given below, they are extracted from the first example (button "Power"). However, as can be seen, all examples present the same similarities even if the final bit patters is different from one button to the other).

- The fact that the timing is different for every step received is most probably due to the imprecision of the remote control unit itself and / or the precision timing error of the Pico-Remote-Analyzer. Whatever the case, we should make an average and consider all values "around 550 microseconds" to BE 550 usec (microseconds).

- The same applies to other values: values around 1675 usec (microseconds) must be considered to BE 1675 usec (microseconds), as well as values around 4450 usec (microseconds).

- There is a value (46340 microseconds) that is really "out-of-scope" (not in the same order of magnitude as other timings). We can think of this as a "separator" between the first part of the command and the second part of the command.

- The first two steps (around 4450 microseconds) are probably some kind of "get ready" or "wake up" signal to announce that a data stream is coming.

- The same two steps (4450 microseconds) also appear after the "separator" and probably announce that another stream is coming. So, the idea of a "get ready" or "wake up" signal still make sense for those two "4450" as well as the first ones.

- All Low level steps are 550 microseconds, whereas High level steps are either 550 microseconds or 1675 microseconds. It also makes sense to think that what distinguishes a 0 bit from a 1 bit is the High level, whereas the Low level is simply for framing purposes (that is, to "force" a level change between different bits).

- So, a pair of steps of 550 usec and 550 usec would represent a "0" bit and a pair of steps of 550 usec and 1675 usec would represent a "1" bit.

- Looking at each part of the stream (before and after the 46340 usec), if we consider that each pair of steps represents one bit, there are 32 bits before and 32 bits after the separator.

NOTE: Even if our deductions are false… For example, even if our "0" bits and "1" bits are the opposite of what they were intended to be, as long as we can decode an infrared data stream and always end up with the same data, whatever the stream may represent, we will always decode the same infrared burst with the same result and be able to trigger the same action. As was said in the introduction of this User Guide, this is the goal of this project.

Based on the observations above, we could try to decode the data stream as follow:

- First, we have two steps (1 and 2) that are a "wake up" signal and announce the beginning of a data stream.

- Then, we consider bits 3 and 4 as the pair of signals representing the first bit of the stream (579 and 1684, or if we "standardize" the timings as previously suggested: 550 and 1675), which translates as a "1" bit.

- Bits 5 and 6, as well as 7 and 8 are also a pair of 550 and 1675 (two other "1" bits).

- Bits 9 and 10, 11 and 12, 13 and 14, 15 and 16, 17 and 18 are pairs of 550 and 550, which would translate as five times "0" bits.

- And so on and so forth…

- If we complete the exercise with all the bits, the resulting bit patterns that we obtain are the following:

NOTE: The column "Data stream (in hex)" has been split into two parts to represent the two values received in the data stream, assuming that the 46340 usec in the middle of the stream splits the stream into two parts.

Button name	**Data stream (in hex)**	
Power	0xE0E040BF	0xE0E040BF
TV	0xE0E0D827	0xE0E0D827
1 <numeral one>	0xE0E020DF	0xE0E020DF
Volume Up	0xE0E0E01F	0xE0E0E01F
Channel Down	0xE0E008F7	0xE0E008F7
Play	0xE0E0E21D	0xE0E0E21D

We can see that there seems to be a "signature" in add data stream: indeed they all begin with "0xE0E0". It is not unusual to see all buttons of a remote control unit having something in common like this. We could think that if most companies use some kind of prefix like this (which is different from one company to another), there is less chance that a remote control from a company will interfere with another remote control from another company.

We can see also that in each data stream, the same value is repeated twice. For example, the button "Power" sends the hex string 0xE0E040BF twice, separated by the pair 550 / 46340 (both streams begin with the "wake up pair" 4450 / 4450. I've seen many remote controls sending the same data twice in the infrared stream like this. The receiving application may then compare both values to make sure they match and this way, validate the integrity of the decoded data stream. Pico-Remote-Analyzer being a generic tool, this specific "match check" has not been added for now, but it could be added by user if wanted.

Some brands will also send the same data twice, but the second time, the bit order is reversed. You may also realize that sometimes, if you keep your finger just a little too long on the button, the data will be repeated three times instead of two.

At this point, we went through most observations / analysis required to complete the exercise. What we need to do now is to adapt the algorithm in Pico-Remote-Analyzer to properly and automatically decode the data stream. This is what will be discussed in the next section.

13) Adapt the Firmware decoding algorithm

Introduction

A quick discussion before going to the code…

The decoding function which is specific to a remote control unit brand and model number is segregated in a stand-alone file which has been given the name of the brand (you may need to add the model number to the brand if you have more than one unit from the same company and if the timing is different between units).

When you started the Firmware earlier, remember there was a brand name suggested by default? This name comes from the include file that was specified the last time the Firmware was built. This include file is named like "Samsung.c", "Memorex.c" or "Panasonic.c", representing the brand name of the remote control unit it is programmed to decode.

So now, we need to adapt the decoding function "decode_ir_command()" so that it is able to decode our remote control unit.

1) Let`s take one of the already existing "brand name file" and copy it under a new name (corresponding to the unit brand name we are decoding), while leaving the original file untouched since it has been done to decode another brand of remote control (for this step, since the remote control unit being analyses is a Samsung unit, this is the command that has been done: "cp Memorex.c Samsung.c").

2) Most items requiring adjustment have been put as "#define" at the top of the Samsung.c file. You may experience cases where you will have to change something else in the code. If ever the case, you should try to make changes in the function "decode_ir_command()" only and keep the rest of the source code (the main module) unchanged, so that the main module remains compatible with all "brand name files" already created. This way, we can simply change the name of the include file in Pico-Remote-Analyzer.c to include the logic and decode a new remote control unit.

3) When we look the burst timing table of Appendix A, we see that there are 32 bits total. When the two "get-ready" steps have been discarded, the step pair 3 and 4 represents bit 1, step pair 5 and 6 represent bit 2, and so on and so forth, until pair 65 and 66 representing bit 32. Pair 67 and 68 represents what we called a "separator" between the first data and its copy. So, we can set the #define NUMBER_OF_BITS as 32 in the Samsung.c file.

4) As we've seen in all our examples, the total number of steps has always been 135. Again, it may happen that you get 203 if you keep your finger a little too long on

the button. This is because the remote will have sent 3 copies of the data instead of the usual 2 copies. If ever the case, simply redo the same button once again to make sure that you receive the usual 135 steps. So, we can also set the #define NUMBER_OF_STEPS as 135 in the Samsung.c file.

5) At the beginning of the timing table, we see that 2 steps of 4450 usec are considered “wake-up steps”. Returning to the Samsung.c file, we can set #define NUMBER_OF_WAKEUP_STEPS as 2.

6) The length of the signal that we called “Separator” is 46340 usec. We see that this length is way off from what we use to see for data bits. The code will recognize a “Separator” when the duration of a “High level” step if equal or longer than the number of microseconds in the #define SEPARATOR. I’ve put 10000 which is much longer than any “1” bits (1675 usec). This way, there will be no problem for the “Separator” of 46340 to be recognized (it is much longer), but at the same time, this eventually opens the door for the same decode_ir_command() function to work fine with another remote control unit (maybe from another brand ?) that would use a similar protocol, but with a shorter Separator length.

7) The last #define to adjust is the #define TRIGGER_POINT_0_1. This represents the length (in microseconds) that will discriminate a 0 bit from a 1 bit. This value will apply only to “High level” steps (since we observed that all “Low level” steps are 550 usec and are useless to determine 0 or 1 bits. Also, we must be careful about one thing: Pico-Remote-Analyzer is a standalone application and nothing else runs on the Pico when you use it. When an interrupt comes in, the Pico will dedicate all its processor time to decode the infrared burst. However, if you add a remote control to one of your own project, you may have callback functions and / or other interrupt service routines running simultaneously. This could bring race conditions where the infrared sensor will not be serviced as fast as it is on the Pico-Remote-Analyzer. You may also find other remote units from the same manufacturer having timing slightly different. For this reason, it is a good idea not to be too strict on the timing. I put 750 as the trigger point between the timing for a “0” bit (550 usec) and a “1” bit (1675 usec). In other words, if a High level is less that 750 usec, it will be decoded as a “0” and if a High level is more than 750 usec, it will be decoded as a “1”.

This completes the modifications required to decode our remote control unit.

NOTE: You should be able to decode many other brands of remote controls by simply adjusting the parameters specified in those “#define” at the top of the “*brand_filename*.c”

14) Decode all remote control buttons

Once you modified the decode_ir_command() function, it is time to test it and proceed to decoding all buttons of the remote control.

At this point, you must have rebuilt and re-flashed the Firmware (Pico-Remote-Analyzer.c) after changing the #define REMOTE_FILENAME to represent the filename of the remote control you analyzed (in step 13.1 above). Restart the Firmware from scratch to make sure the internal button list is empty. Also make sure the log file is active on the terminal emulator program.

Confirm that the default brand name of the remote control unit proposed by the Firmware is correct (it should be Ok since it is derived from the include filename).

Now, proceed to the decoding of all remote control buttons (I suggest you proceed from top to bottom to make sure you do not forget any button).

Press a remote button, then select the menu choice: "Decode this infrared burst using file: "*brand_filename*.c" (where "*brand_filename*.c" should correspond to the filename in which you adapted the decode_ir_command() function).

You will be asked to enter the button name and then, all timing information, along with decoding results will be sent to log file. You will be ask to enter "x" ("capital X" also works) to save this button data. If the result of the decoding seems normal, simply press "x" and the button name will be temporarily saved internally, along with its corresponding command value as it has been decoded (in hex). If the result of the decoding seems strange, you simply skip the "press x" (press <enter> instead) and redo the same button again.

Proceed like this with all remote buttons and then proceed to next step.

15) Display all remote control buttons and commands

When you have decoded all remote control buttons and save them by pressing “x” when asked to do so, it is time to generate the list of all buttons decoded, along with their command value (in hex). You first need to press a remote button again (so that the menu appears… any button will work). When you have access to the menu, select: “Display complete remote control button list”. All decoding results for which you pressed “x” will be displayed and saved to the log file. See an example in Appendix B.

Keep the log file for future reference and give it a name with the brand and model number.

16) Integrate a remote control in a program

Now that you decoded your remote control and understand all the underlying timings, you may want to integrate it to one of your projects based on a Pico. Simply follow those steps:

- On a sheet of paper, give a specific function ("action" in your project) and a name to all remote control buttons that you plan to use. (for example: Function: "turn On red LED when an alarm occurs and Name: "IR_TURN_ON_RED_LED"

- When you decided all functions required and gave them a specific name, edit the include file created for your remote control unit (for example, by copying the "Samsung.h" file to "*your_filename*.h") and enter the list of all #define IR_XXX for each function / button that you listed in the previous step.

- Add this include file in your main project file and erase the corresponding #include that is now in the "*your_filename*.c" file. (We need to make those #define global to the whole project now… not only local to the decode_id_command() function).

- In the "*your_filename*.c" file, there is a "return" instruction that is currently inserted about one-third deep in the file, just before the "switch (DataBuffer)" instruction. We need to execute the switch statement that will select and return the hex value decoded in the first part of the function. So, the "return" instruction must be erased.

- Below the "return" instruction that you just removed, you must replace all generic "IR_COMMAND_TO_EXECUTE" with the function names you wrote in the "*your_filename*.h" and corresponding to each remote control button.

- Next steps must be done in your project source code. You first need to set an interrupt service routine ("ISR") to receive the infrared data stream. You can simply copy-paste the one from Pico-Remote-Analyzer.

- Most embedded programs consist of an endless loop. You must add a check in this loop to verify the current value of IrSteps. If the value is different than zero, it means that an infrared data stream has been received. If this is the case, add a quick delay (to make sure all infrared steps have been received), then call the "decode_ir_command()" function. This function will return the IR_XXX command you allocated to the remote control button. You can then take action for the target function.

- Once again, you can refer to the Firmware Pico-Green-Clock in my repository for an example of remote control integration. Pico-Remote-Analyzer has been fine-tuned *after* Pico-Green-Clock, so you may see differences between both, but nonetheless, you should be able to find your way.

Appendix A – Infrared burst timing analysis

This Appendix contains infrared burst timing examples as they have been received from a Samsung remote control unit.

Refer to section 12 above for observations / analysis about those examples.

As explained in section 12 above, remote control unit details are given in the header. The remote control button appears just below the header. For this specific remote control unit, 135 timing steps are generated for each button, so two pages are required to cover the complete infrared data stream for each button.

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          Flash-Remote-Analyzer
                       Microcontroller is a Pico W
                  Pico's Unique ID: E661-4103-E776-2623
                      Brand under analysis: Samsung
                Remote control model number: BN59-00673A
                             Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: Power
```

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4449	51	low	582
2	high	4457	52	high	1667
3	low	579	53	low	576
4	high	1684	54	high	547
5	low	558	55	low	580
6	high	1685	56	high	1669
7	low	558	57	low	575
8	high	1687	58	high	1667
9	low	560	59	low	578
10	high	547	60	high	1668
11	low	584	61	low	575
12	high	567	62	high	1675
13	low	557	63	low	571
14	high	549	64	high	1666
15	low	582	65	low	578
16	high	546	66	high	1667
17	low	583	67	low	580
18	high	544	68	high	46340
19	low	584	69	low	4458
20	high	1685	70	high	4475
21	low	562	71	low	577
22	high	1684	72	high	1667
23	low	558	73	low	581
24	high	1687	74	high	1665
25	low	559	75	low	581
26	high	545	76	high	1666
27	low	584	77	low	579
28	high	546	78	high	549
29	low	579	79	low	578
30	high	548	80	high	551
31	low	585	81	low	579
32	high	536	82	high	551
33	low	602	83	low	579
34	high	515	84	high	549
35	low	585	85	low	579
36	high	543	86	high	550
37	low	582	87	low	579
38	high	1687	88	high	1666
39	low	557	89	low	578
40	high	547	90	high	1666
41	low	583	91	low	579
42	high	547	92	high	1666
43	low	581	93	low	577
44	high	546	94	high	552
45	low	580	95	low	579
46	high	548	96	high	549
47	low	585	97	low	577
48	high	543	98	high	552
49	low	581	99	low	579
50	high	547	100	high	551

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          Flash-Remote-Analyzer
                       Microcontroller is a Pico W
                  Pico's Unique ID: E661-4103-E776-2623
                      Brand under analysis: Samsung
                Remote control model number: BN59-00673A
                             Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: Power

 Step      Logic    Duration                Step      Logic    Duration
number     level                           number     level

  101        low        577
  102       high        551
  103        low        577
  104       high        551
  105        low        577
  106       high       1667
  107        low        577
  108       high        551
  109        low        573
  110       high        555
  111        low        576
  112       high        553
  113        low        573
  114       high        555
  115        low        573
  116       high        554
  117        low        575
  118       high        555
  119        low        573
  120       high       1672
  121        low        572
  122       high        556
  123        low        570
  124       high       1675
  125        low        570
  126       high       1676
  127        low        543
  128       high       1700
  129        low        570
  130       high       1675
  131        low        568
  132       high       1677
  133        low        543
  134       high       1703
  135        low        543
  136        ---          0
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

= =

Flash-Remote-Analyzer
Microcontroller is a Pico W
Pico's Unique ID: E661-4103-E776-2623
Brand under analysis: Samsung
Remote control model number: BN59-00673A
Step count: 135

= =

Button: TV

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4434	51	low	581
2	high	4486	52	high	572
3	low	578	53	low	533
4	high	1684	54	high	567
5	low	557	55	low	583
6	high	1684	56	high	1690
7	low	558	57	low	558
8	high	1687	58	high	571
9	low	556	59	low	531
10	high	571	60	high	598
11	low	531	61	low	526
12	high	598	62	high	1717
13	low	531	63	low	556
14	high	599	64	high	1688
15	low	531	65	low	559
16	high	597	66	high	1686
17	low	530	67	low	532
18	high	576	68	high	46294
19	low	582	69	low	4462
20	high	1686	70	high	4471
21	low	531	71	low	580
22	high	1714	72	high	1689
23	low	554	73	low	557
24	high	1691	74	high	1688
25	low	561	75	low	558
26	high	543	76	high	1687
27	low	556	77	low	556
28	high	574	78	high	548
29	low	580	79	low	581
30	high	546	80	high	548
31	low	555	81	low	554
32	high	576	82	high	576
33	low	552	83	low	582
34	high	599	84	high	546
35	low	558	85	low	554
36	high	1686	86	high	574
37	low	557	87	low	579
38	high	1688	88	high	1690
39	low	558	89	low	531
40	high	550	90	high	1716
41	low	577	91	low	549
42	high	1690	92	high	1678
43	low	556	93	low	553
44	high	1688	94	high	548
45	low	555	95	low	580
46	high	553	96	high	547
47	low	578	97	low	578
48	high	572	98	high	552
49	low	532	99	low	552
50	high	572	100	high	575

= =

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          Flash-Remote-Analyzer
                       Microcontroller is a Pico W
                  Pico's Unique ID: E661-4103-E776-2623
                      Brand under analysis: Samsung
                Remote control model number: BN59-00673A
                             Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: TV

 Step      Logic    Duration                Step      Logic    Duration
number     level                           number     level

  101        low        578
  102       high        559
  103        low        571
  104       high       1691
  105        low        552
  106       high       1693
  107        low        551
  108       high        552
  109        low        580
  110       high       1693
  111        low        524
  112       high       1719
  113        low        550
  114       high        553
  115        low        554
  116       high        573
  117        low        574
  118       high        554
  119        low        549
  120       high        579
  121        low        576
  122       high        554
  123        low        550
  124       high       1695
  125        low        575
  126       high        554
  127        low        549
  128       high        578
  129        low        542
  130       high       1706
  131        low        575
  132       high       1668
  133        low        576
  134       high       1669
  135        low        574
  136        ---          0
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                         Flash-Remote-Analyzer
                       Microcontroller is a Pico W
                  Pico's Unique ID: E661-4103-E776-2623
                      Brand under analysis: Samsung
                Remote control model number: BN59-00673A
                             Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: <1>
```

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4463	51	low	559
2	high	4457	52	high	1684
3	low	587	53	low	558
4	high	1674	54	high	1687
5	low	564	55	low	558
6	high	1681	56	high	571
7	low	559	57	low	557
8	high	1685	58	high	1688
9	low	563	59	low	559
10	high	567	60	high	1686
11	low	561	61	low	557
12	high	569	62	high	1688
13	low	558	63	low	559
14	high	570	64	high	1687
15	low	563	65	low	562
16	high	565	66	high	1682
17	low	538	67	low	558
18	high	569	68	high	46335
19	low	585	69	low	4468
20	high	1683	70	high	4463
21	low	564	71	low	587
22	high	1680	72	high	1684
23	low	536	73	low	559
24	high	1711	74	high	1687
25	low	556	75	low	531
26	high	551	76	high	1713
27	low	581	77	low	560
28	high	570	78	high	544
29	low	560	79	low	583
30	high	568	80	high	548
31	low	561	81	low	579
32	high	569	82	high	547
33	low	562	83	low	583
34	high	566	84	high	547
35	low	561	85	low	580
36	high	567	86	high	547
37	low	561	87	low	583
38	high	543	88	high	1686
39	low	586	89	low	560
40	high	1681	90	high	1686
41	low	561	91	low	556
42	high	568	92	high	1688
43	low	559	93	low	558
44	high	570	94	high	547
45	low	558	95	low	586
46	high	570	96	high	542
47	low	560	97	low	582
48	high	547	98	high	546
49	low	582	99	low	579
50	high	569	100	high	549

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                           Flash-Remote-Analyzer
                        Microcontroller is a Pico W
                   Pico's Unique ID: E661-4103-E776-2623
                       Brand under analysis: Samsung
                 Remote control model number: BN59-00673A
                              Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: <1>

 Step      Logic    Duration                   Step      Logic    Duration
number     level                              number     level

  101        low        579
  102       high        549
  103        low        582
  104       high        546
  105        low        580
  106       high        550
  107        low        582
  108       high       1691
  109        low        549
  110       high        551
  111        low        579
  112       high        552
  113        low        583
  114       high        544
  115        low        578
  116       high        552
  117        low        579
  118       high        549
  119        low        579
  120       high       1662
  121        low        569
  122       high       1664
  123        low        577
  124       high        549
  125        low        575
  126       high       1672
  127        low        574
  128       high       1669
  129        low        576
  130       high       1670
  131        low        576
  132       high       1667
  133        low        579
  134       high       1666
  135        low        576
  136        ---          0
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                       Flash-Remote-Analyzer
                    Microcontroller is a Pico W
               Pico's Unique ID: E661-4103-E776-2623
                   Brand under analysis: Samsung
             Remote control model number: BN59-00673A
                          Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: Volume Up
```

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4457	51	low	551
2	high	4465	52	high	575
3	low	547	53	low	580
4	high	1714	54	high	548
5	low	534	55	low	552
6	high	1711	56	high	577
7	low	555	57	low	556
8	high	1691	58	high	1714
9	low	557	59	low	553
10	high	548	60	high	1692
11	low	551	61	low	553
12	high	600	62	high	1692
13	low	527	63	low	527
14	high	601	64	high	1717
15	low	531	65	low	554
16	high	575	66	high	1692
17	low	550	67	low	526
18	high	602	68	high	46366
19	low	528	69	low	4432
20	high	1715	70	high	4499
21	low	555	71	low	552
22	high	1690	72	high	1719
23	low	528	73	low	525
24	high	1715	74	high	1721
25	low	533	75	low	550
26	high	576	76	high	1695
27	low	551	77	low	529
28	high	599	78	high	575
29	low	528	79	low	552
30	high	575	80	high	570
31	low	554	81	low	544
32	high	576	82	high	571
33	low	551	83	low	551
34	high	577	84	high	578
35	low	553	85	low	548
36	high	1714	86	high	581
37	low	555	87	low	550
38	high	1690	88	high	1699
39	low	555	89	low	544
40	high	1691	90	high	1699
41	low	530	91	low	547
42	high	575	92	high	1697
43	low	552	93	low	547
44	high	577	94	high	580
45	low	551	95	low	518
46	high	576	96	high	609
47	low	578	97	low	541
48	high	546	98	high	587
49	low	581	99	low	519
50	high	553	100	high	612

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = 
                       Flash-Remote-Analyzer
                    Microcontroller is a Pico W
               Pico's Unique ID: E661-4103-E776-2623
                   Brand under analysis: Samsung
             Remote control model number: BN59-00673A
                          Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = 
Button: Volume Up

 Step      Logic    Duration                 Step      Logic    Duration
number     level                            number     level

  101        low        516
  102       high        611
  103        low        517
  104       high       1730
  105        low        546
  106       high       1698
  107        low        547
  108       high       1698
  109        low        546
  110       high        581
  111        low        539
  112       high        590
  113        low        540
  114       high        589
  115        low        538
  116       high        589
  117        low        540
  118       high        588
  119        low        540
  120       high        588
  121        low        540
  122       high        588
  123        low        540
  124       high        588
  125        low        540
  126       high       1706
  127        low        548
  128       high       1699
  129        low        545
  130       high       1698
  131        low        546
  132       high       1700
  133        low        546
  134       high       1698
  135        low        547
  136        ---          0
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = 
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                        Flash-Remote-Analyzer
                     Microcontroller is a Pico W
                Pico's Unique ID: E661-4103-E776-2623
                    Brand under analysis: Samsung
              Remote control model number: BN59-00673A
                           Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: Channel Down
```

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4455	51	low	550
2	high	4467	52	high	1720
3	low	547	53	low	527
4	high	1717	54	high	1719
5	low	551	55	low	524
6	high	1693	56	high	1721
7	low	528	57	low	526
8	high	1718	58	high	1720
9	low	529	59	low	524
10	high	577	60	high	577
11	low	549	61	low	552
12	high	602	62	high	1720
13	low	526	63	low	526
14	high	577	64	high	1720
15	low	551	65	low	551
16	high	577	66	high	1694
17	low	578	67	low	524
18	high	551	68	high	46358
19	low	552	69	low	4447
20	high	1717	70	high	4468
21	low	526	71	low	577
22	high	1719	72	high	1691
23	low	529	73	low	525
24	high	1716	74	high	1719
25	low	528	75	low	524
26	high	575	76	high	1720
27	low	552	77	low	525
28	high	578	78	high	577
29	low	551	79	low	550
30	high	576	80	high	578
31	low	579	81	low	548
32	high	549	82	high	578
33	low	552	83	low	549
34	high	579	84	high	580
35	low	548	85	low	549
36	high	577	86	high	578
37	low	553	87	low	549
38	high	577	88	high	1699
39	low	549	89	low	546
40	high	576	90	high	1723
41	low	577	91	low	522
42	high	553	92	high	1699
43	low	553	93	low	546
44	high	1717	94	high	579
45	low	530	95	low	572
46	high	574	96	high	556
47	low	551	97	low	541
48	high	578	98	high	588
49	low	553	99	low	540
50	high	575	100	high	590

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                            Flash-Remote-Analyzer
                         Microcontroller is a Pico W
                    Pico's Unique ID: E661-4103-E776-2623
                        Brand under analysis: Samsung
                  Remote control model number: BN59-00673A
                               Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: Channel Down

 Step      Logic    Duration                Step      Logic    Duration
number     level                           number     level

  101        low        517
  102       high        612
  103        low        516
  104       high        611
  105        low        540
  106       high        589
  107        low        538
  108       high        589
  109        low        540
  110       high        589
  111        low        539
  112       high       1706
  113        low        547
  114       high        581
  115        low        540
  116       high        588
  117        low        543
  118       high        584
  119        low        541
  120       high       1705
  121        low        547
  122       high       1698
  123        low        545
  124       high       1699
  125        low        546
  126       high       1698
  127        low        547
  128       high        583
  129        low        539
  130       high       1706
  131        low        570
  132       high       1675
  133        low        571
  134       high       1675
  135        low        546
  136        ---          0

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

= =

Flash-Remote-Analyzer
Microcontroller is a Pico W
Pico's Unique ID: E661-4103-E776-2623
Brand under analysis: Samsung
Remote control model number: BN59-00673A
Step count: 135

= =

Button: <Play>

Step number	Logic level	Duration	Step number	Logic level	Duration
1	low	4464	51	low	559
2	high	4456	52	high	545
3	low	584	53	low	558
4	high	1680	54	high	573
5	low	565	55	low	583
6	high	1678	56	high	567
7	low	558	57	low	533
8	high	1687	58	high	1713
9	low	563	59	low	560
10	high	564	60	high	1684
11	low	559	61	low	560
12	high	569	62	high	1686
13	low	562	63	low	560
14	high	567	64	high	570
15	low	562	65	low	562
16	high	567	66	high	1682
17	low	560	67	low	559
18	high	569	68	high	46373
19	low	560	69	low	4494
20	high	1685	70	high	4437
21	low	560	71	low	587
22	high	1685	72	high	1684
23	low	562	73	low	561
24	high	1683	74	high	1683
25	low	559	75	low	558
26	high	569	76	high	1687
27	low	559	77	low	560
28	high	571	78	high	545
29	low	561	79	low	578
30	high	567	80	high	574
31	low	560	81	low	586
32	high	570	82	high	519
33	low	559	83	low	584
34	high	569	84	high	567
35	low	561	85	low	558
36	high	1686	86	high	546
37	low	559	87	low	560
38	high	1684	88	high	1710
39	low	536	89	low	560
40	high	1709	90	high	1686
41	low	559	91	low	557
42	high	570	92	high	1688
43	low	560	93	low	556
44	high	545	94	high	550
45	low	585	95	low	583
46	high	567	96	high	546
47	low	559	97	low	584
48	high	1686	98	high	542
49	low	562	99	low	584
50	high	567	100	high	570

= =

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                        Flash-Remote-Analyzer
                     Microcontroller is a Pico W
                Pico's Unique ID: E661-4103-E776-2623
                    Brand under analysis: Samsung
              Remote control model number: BN59-00673A
                           Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Button: <Play>

 Step      Logic    Duration                Step      Logic    Duration
number     level                           number     level

  101        low        558
  102       high        547
  103        low        582
  104       high       1688
  105        low        560
  106       high       1686
  107        low        561
  108       high       1685
  109        low        560
  110       high        545
  111        low        591
  112       high        539
  113        low        583
  114       high        544
  115        low        555
  116       high       1714
  117        low        555
  118       high        549
  119        low        586
  120       high        542
  121        low        581
  122       high        546
  123        low        583
  124       high        548
  125        low        582
  126       high       1687
  127        low        559
  128       high       1687
  129        low        559
  130       high       1686
  131        low        559
  132       high        544
  133        low        584
  134       high       1687
  135        low        555
  136        ---          0

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

Appendix B – Display all decoded buttons and commands

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          Flash-Remote-Analyzer
                       Microcontroller is a Pico W
                  Pico's Unique ID: E661-4103-E776-2623
                      Brand under analysis: Samsung
                Remote control model number: BN59-00673A
                             Step count: 135
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Number of buttons decoded: 47

      Remote control            Infrared command
         button name                     decoded

[  1]          Power                  0xE0E040BF
[  2]             TV                  0xE0E0D827
[  3]              1                  0xE0E020DF
[  4]              2                  0xE0E0A05F
[  5]              3                  0xE0E0609F
[  6]              4                  0xE0E010EF
[  7]              5                  0xE0E0906F
[  8]              6                  0xE0E050AF
[  9]              7                  0xE0E030CF
[ 10]              8                  0xE0E0B04F
[ 11]              9                  0xE0E0708F
[ 12]              0                  0xE0E08877
[ 13]              -                  0xE0E0C43B
[ 14]         Pre-Ch                  0xE0E0C837
[ 15]           Mute                  0xE0E0F00F
[ 16]         Source                  0xE0E0807F
[ 17]      Volume Up                  0xE0E0E01F
[ 18]    Volume Down                  0xE0E0D02F
[ 19]     Channel Up                  0xE0E048B7
[ 20]   Channel Down                  0xE0E008F7
[ 21]           Menu                  0xE0E058A7
[ 22]        Ch List                  0xE0E0D629
[ 23]        W. Link                  0xE0E031CE
[ 24]          Tools                  0xE0E0D22D
[ 25]         Return                  0xE0E01AE5
[ 26]           Info                  0xE0E0F807
[ 27]           Exit                  0xE0E0B44B
[ 28]           <Up>                  0xE0E006F9
[ 29]         <Down>                  0xE0E08679
[ 30]         <Left>                  0xE0E0A659
[ 31]        <Right>                  0xE0E046B9
[ 32]        <Enter>                  0xE0E016E9
[ 33]          <Red>                  0xE0E036C9
[ 34]        <Green>                  0xE0E028D7
[ 35]       <Yellow>                  0xE0E0A857
[ 36]         <Blue>                  0xE0E06897
[ 37]             CC                  0xE0E0A45B
[ 38]            MTS                  0xE0E000FF
[ 39]            DMA                  0xE0E0C639
[ 40]         E.Mode                  0xE0E029D6
[ 41]        P. Size                  0xE0E07C83
[ 42]       Fav. Ch.                  0xE0E022DD
[ 43]       <Rewind>                  0xE0E0A25D
[ 44]        <Pause>                  0xE0E052AD
[ 45]      <Forward>                  0xE0E012ED
[ 46]         <Play>                  0xE0E0E21D
[ 47]         <Stop>                  0xE0E0629D
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

Appendix C – List of GPIOs in Pico-Remote-Analyzer

GPIO number	Direction / Usage	Description
GPIO 0	(Out)	Pico’s UART output to an external terminal emulator.
GPIO 1	(In)	Pico’s UART input from an external terminal emulator program.
GPIO 2		Not used.
GPIO 3		Not used.
GPIO 4		Not used.
GPIO 5		Not used.
GPIO 6		Not used.
GPIO 7		Not used.
GPIO 8		Not used.
GPIO 9		Not used.
GPIO 10		Not used.
GPIO 11		Not used.
GPIO 12		Not used.
GPIO 13		Not used.
GPIO 14		Not used.
GPIO 15		Not used.
GPIO 16		Not used.
GPIO 17		Not used.
GPIO 18		Not used.
GPIO 19		Not used.
GPIO 20		Not used.
GPIO 21		Not used.
GPIO 22	(In)	VS1838b infrared receiver for remote control
GPIO 23		Used internally for voltage regulation
GPIO 24		Used internally for voltage detection
GPIO 25	(Out)	On-board Pico`s LED (doesn’t work as usual on Pico W).
GPIO 26		Not used.
GPIO 27		Active buzzer
GPIO 28		Not used.
GPIO 29		ADC-Vref (Power supply voltage reading)