Building the Assistive Technology Ultimate Remote

Created by Chris Young

Last updated on 2019-09-23 09:29:39 PM UTC
Overview

In my lifelong struggle to deal with my neuromuscular disease Spinal Muscular Atrophy (SMA), I've built a variety of assistive technology devices. But the most powerful and useful device I've built is my "Ultimate Remote". It combines infrared control of TV, DVR/cable box, DVD/Blu-ray players as well as IR mouse and keyboard emulation. It also includes a Bluetooth device for iOS switch control of my iPhone or iPad. The first version of this device was created in late 2015 and it has evolved through several versions. It's always been my intent to document the device so that other disabled people can take advantage of what I've developed. Because it was an evolving design, I've not documented it until now. However, I've cleaned up the code, redesigned the 3D printed enclosure, and it's finally ready to share with the world.

This is a very complicated and sophisticated device that performs a wide variety of functions. This tutorial will be very long and requires significant maker skills including soldering, wiring, 3D printing, and a willingness to learn a lot about infrared device control and some Bluetooth functions. While it isn't necessary to know any Arduino programming in C++, you should be familiar with how to connect the board to the Arduino IDE, modify a sketch, and upload it.

Adaptations and Support

One of the main advantages of maker-built assistive technology is that it can be adapted to a particular user's needs. This tutorial describes everything that I have put into my ultimate remote. You may need to modify or adapt it to your particular needs. I will describe some options along the way but for the most part this tutorial will describe the device that I build for myself and you will have to make your own modifications or adaptations. Feel free to contact me for advice especially when it comes to configuring the infrared controls for your TV and other devices.

Normally I could not give individual support to readers of these tutorials however for those that will use this device, it has the potential to have a great impact in their independence and ability to overcome their handicaps. The device has been a lifesaver for me literally. I used the Bluetooth switch control to communicate with doctors and nurses when I could not speak well on a ventilator in the hospital. All of the Fusion 360 design work, all of the programming, and all of the writing of this tutorial including everything the videos was done with the assistance of this Ultimate Remote and dictation software Dragon NaturallySpeaking. That should illustrate how important this device is to people like me.

If you need assistance building or configuring the device feel free to contact me at cy_borg5@cyborg5.com or visit Facebook at the ATMakers Facebook page (https://adafruit.it/FMj). We will give you as much help as we can with this or other assistive technology issues.
Composition

The device includes an LCD TFT display, an M0 Feather BLE SAMD21 processor, a Li-Ion battery and a Power Boost 1000c. The battery and Power Boost can be used not only to power the device but also provide supplemental power to an iPhone. We've included a barrel jack to make recharging easier so you don't wear out the micro B USB connector on the Power Boost (which I've done).

Note: You cannot recharge the battery using the micro USB on the feather board itself. You can only recharge it using the micro USB on the power boost or through the barrel jack.

Receiver and Building Block Tutorials

If you're going to use the device to control mouse and keyboard functions on a PC or laptop, you will also need to construct a receiver dongle using an Adafruit Trinket M0 and an IR receiver chip.

The device uses an open source IR transmitter board that I developed. Complete instructions on how to build that board are available in a separate tutorial here.

- Building an Infrared Transmitter and Receiver Board([https://adafruit.it/FM3](https://adafruit.it/FM3))

The IR transmit and receive functions are handled by an open source library called IRLib2. Here is a tutorial on how to use that library.

- Using an Infrared Library on Arduino([https://adafruit.it/vxc](https://adafruit.it/vxc))

There is also an extensive user's manual available with the library and I am available to answer individual questions about its use.

The iOS switch control functions are explained in more detail in this tutorial.

- Using Bluefruit BLE to Give Disabled Users Switch Control Access to iOS Devices([https://adafruit.it/FM4](https://adafruit.it/FM4))

Project Genesis

You might want to make yourself familiar with all of the above before embarking on this project.

If you are interested in the background and evolution of this device, I have some blog entries that document my various struggles to operate IR devices and how the Ultimate Remote came into being.

- The Ultimate Remote and Why I Built It([https://adafruit.it/FM5](https://adafruit.it/FM5))
- The Ultimate Remote Is Dead. Long Live the Ultimate Remote 2.0([https://adafruit.it/FM6](https://adafruit.it/FM6))
- Taking A Tour of My Assistive Technology([https://adafruit.it/FM7](https://adafruit.it/FM7))

The last link is a video that shows a variety of assistive technology devices I use around my house but a big portion of it is devoted to demonstrating the use of the Ultimate Remote. If you have questions you can email me at cy_borg5@cyborg5.com.
Parts List

For the main processor, you will need a Feather M0 Bluefruit BLE. This will allow you to do switch control on an iOS device such as iPhone or iPad. If for some reason you choose not to do the BLE switch control, you could substitute a different Feather M0 or M4, however IRLib does not support nRF52832 or nRF52840 devices, so that is not a viable option.

![Adafruit Feather M0 Bluefruit LE](image)

Adafruit Feather M0 Bluefruit LE

$29.95
IN STOCK
Add To Cart

For display, the 2.4 inch TFT FeatherWing is used.

![TFT FeatherWing - 2.4" 320x240 Touchscreen For All Feathers](image)

TFT FeatherWing - 2.4" 320x240 Touchscreen For All Feathers

$29.95
IN STOCK
Add To Cart

Although the Feather has a built-in charger, we decided not to use it. Instead we are including a Power Boost 1000c. This allows you to keep an iPhone or iPad charged using the remote's battery. It’s not sufficiently powerful enough to charge a dead iPhone or iPad while simultaneously running the remote, but it does keep the device topped up if it is already charged. On a normal day, I have no trouble going all day without recharging. If I am going to do live streaming on the phone or shooting lots of video, I sometimes need to recharge or plug-in a supplemental battery.
To preserve your battery life if you are not using the device for a long period and not charging it, you should have a power switch.

In my day-to-day usage of the device, I plug it in every night. After a while, the micro USB B connector on the power boost failed even though it is one of the most sturdy micro USB connectors I've ever seen. To make it easier to charge the device, we include a 2.1 mm barrel jack as an option.

Warning: The device can be recharged through the micro USB of the power boost or or as a barrel Jack but not through the micro USB of the Feather M0 BLE. When charging through the barrel jack only use chargers that output 5 volts. We recommend this one.
This 4400mAh Li-Ion battery will provide plenty of power to operate the remote all day long and to keep your iPhone or iPad topped up.

We will assemble the case using M3 brass heat set inserts such as these below. You also need four M3 screws. We recommend flat head Phillips screws but rounded screws of any kind would work as well as long as they are M3 size. They are available from a variety of sources including your local hardware store or from this or other online suppliers.

- **M3 Machine Screw 4 mm length, flat head Phillips drive from Fastenal** (https://adafruit.it/FM8)

These heat-set inserts will be melted into the 3D printed plastic case. We recommend you obtain one of these adapters for your soldering iron that will make it easier to press these inserts into place.
Although we prefer the brass inserts and the M3 screws, we also provide an alternative case that can be fastened together with #6-20 sheet metal screws which can be obtained from hardware stores or the source below. When not using the brass inserts and the M3 screws, we prefer sheet metal screws over machine screws because they have a pointed tip that taps into the plastic rather well.

- #6-20 Sheet Metal Screws Phillips flat head from Fastenal (https://adafru.it/FM9)

The TFT display mounts into the case using four 2.5 mm screws. It could also be attached to a case using hot glue or CA glue. Additionally, you will need glue to attach the 2.1 mm barrel jack into the case. We will also secure the battery in the case using tape or hot glue.

The case is designed with a 1/4"-20 T-nut that makes it easy to mount the remote on any 1/4"-20 camera mount or other such bracket. We use one that is 3/4" diameter with 4 prongs. Here are 2 sources or you may find them at your local hardware store. Alternatively you can come up with your own mounting system. Mine is mounted with Velcro onto another bracket.

- T-nut inserts 1/4"-20 threads, 3/4" flange, 4 prongs from McMaster-Carr (https://adafru.it/FMa)
- T-nut inserts 1/4"-20 threads, 3/4" flange, 4 prongs from Fastenal (https://adafru.it/FMb)

The Receiver

If you’re going to use the device to emulate mouse or keyboard on a PC or laptop, you will need to create a receiver dongle using a Trinket M0.

Adafruit Trinket M0 - for use with CircuitPython & Arduino IDE

OUT OF STOCK

You also need a TSOP infrared receiver such as this one for the dongle.
You will also need an IR transmitter board which is described separately in the link below. It will have its own parts list. The IR board has a number of assembly options so you should also see the section in this tutorial titled Infrared Board Assembly (https://adafru.it/FMc) for more details.

- Infrared Transmit and Receive Breakout Board Learning Guide (https://adafru.it/FMd)

Connections to Other Devices

A variety of techniques can be used to activate this device. In this tutorial, we will describe how to connect 3 assistive technology (AT) switches using standard 3.5 mm (1/8 inch) mono jacks. You can obtain such jacks at this link.

- 3.5 mm Panel Mount Mono Jack from DigiKey (https://adafru.it/FMe)

Wiring it all up

Finally, you will need connecting wire. A soft silicone stranded wire will be handy to use in most instances, however some of the wires will have to be solid-core because they need to plug-in to the socket of the TFT display board. Alternatively, you can use a soft silicone wire and solder on a piece of stiff wire or a single pin header. It's good to use a variety of colors so that you can keep things sorted out while you are working on the device and troubleshoot any mistakes easily.
Assembly Instructions

3D Printed Case

We have designed a 3D printed case that will hold all of the parts for this project. We have included not only STL files that can be printed directly we are also including a Fusion 360 design file so that you can adapt the enclosure to your own specific needs. Here is a link to GitHub repository that contains all of the design files as well as the sample software we will use.

Step by Step Instructions

In the pages that follow, you will find step-by-step instructions for how to assemble and wire the device(s). Also below is a YouTube video that shows how to wire up the case and assemble it. There is a separate video for building the receiver dongle.
Infrared Board Assembly

Detailed assembly instructions for the infrared board are given in its own tutorial (https://adafruit.it/FM3). However, there are some options you need to consider. If you're not using the Ultimate Remote to emulate mouse or keyboard on a PC or laptop, then you will not be building a receiver dongle. That means you need to install a receiver chip on the infrared board. There is a place to install 2 receiver chips on the board. One is called a TSOP receiver on the right and the other is a TSMP learner on the left. If you're building a dongle, you will not need to install either of them. But if you're not building a dongle, you will need to install the TSOP on the right in the orientation shown in this image.

The infrared board also has the option of 2 current limiting resistors of 33 ohms each. We recommend not using these resistors and instead installing a jumper wire at position R2 and R3 as shown above. You will still need the 1K ohm resistor as well as the 3 transistors and 2 IR LEDs. We will not be soldering any headers into the board but will put wires directly into the appropriate holes and solder them in place.

We typically use red wires for power and green for ground wires but you can use any color you want. You can use stranded flexible silicone coated wire for the power connections, however the LED wire and the receiver wire must be solid core wire or have a solid tip soldered onto them because they will be plugged into the socket on the TFT display board.

You should solder a red wire anywhere in the column of holes labeled +5v which is the second pin from the left corner. It should be about 4 inches long. Solder a green wire about 4 inches long to the GND hole. You should solder another wire (we used yellow) about 3 inches long to the LED pin hole. It is the fourth from the left. This must be a solid core wire because it is going to plug into the TFT board sockets.
The left image below shows this wiring.

Note: Technically wiring standards specify that you use black wires for ground wires but I’ve always used green. Green for ground is easy to remember and helped my father wire up lots of gadgets over the years. The green shows up well in these illustrations but if you want to be technically correct, you should probably use black wire for ground. Then again it’s your device. Use what makes sense for you.

If you are not building a receiver dongle, there are some additional wires you will have to add to use the receiver on the IR board. These 2 additional wires will have to be solid-core because they will plug into the TFT display. Make them about 3 inches long. The brown wire shown in the right image above connects anywhere along the first column of holes labeled +3.3. The white wire connects to the Recv pin hole which is the fifth from the left.
3D Printed Case

In the GitHub archive, you will find several STL files for 3D printing. There is also a Fusion 360 archive containing the entire design including board placement and wiring diagrams.

In the STL files folder, you will find 2 subfolders. One of them contains parts if you're going to assemble the case using M3 screws and brass inserts that are heat set into the plastic. The other folder is for a case design using #6-20 sheet metal screws that tap into the plastic case.

Within these folders, you also have other options. For example, in the M3 folder you will find case_m3.stl which is the main case and then two versions of the lid. One of them has a hole over the area of the IR receiver. If you are using the mouse and keyboard emulation and building a receiver dongle, then you don't need a hole for the receiver in the remote. Use lid_without_hole_m3.stl. If you are not building a receiver dongle, then you will need to put a receiver chip in the IR board and you will need a hole to access that, so use lid_with_hole_m3.stl.

Similarly, in the folder named 6-20 screws into plastic you will find files case_6-20.stl, lid_with_hole_6-20.stl, and lid_without_hole_6-20.stl to be used if you're assembling the case using #6-20 sheet metal screws.

You should print also door.stl, and guard.stl.

You may insert the brass heat set anchors into the main part of the case in four locations using a soldering iron and a heat set attachment. You should do this before mounting any of the boards. That way if you damage the case, you can reprint it. The small door piece should slide into the receptacle at the end of the case. You might have to file it a little bit with an emery board or fingernail file. There is a raised rib along one edge. That edge should go last. It will give you some grip to remove the door later. The notch goes at the bottom so that you could later pry it open with a screwdriver if the door gets stuck.

If you're building the dongle you should also print dongle_case.stl and dongle_lid.stl. All of the parts are provided in their proper print orientation. They can be printed without supports. We used PLA filament, but if you want extra strength, you could use PETG. We do not recommend ABS because it will shrink and warp and the electronic parts will not properly fit. Because different 3D printers have different tolerances, you may need to open the Fusion 360 file and make some adjustments. You can go into the user parameters and change Gap from 0.2 mm to whatever value best suits your printer's tolerances.
The image below shows the pieces for the dongle. They fit together using a snap fit mechanism and do not require screws.

Customizing the Case

Of course you free to design your own customized 3D printing case if you don't want to use ours. Also there are some options in the Fusion 360 model that will let you make some minor modifications to the case that we provided. If you open up the User Parameters dialogue on Fusion 360 you might want to adjust the following parameters:

- **Shell**: 1.6 mm The thickness of the walls throughout the case.
- **Gap**: 0.2 mm Tolerance between adjacent parts.
- **Suppress_IR_Recv_Hole**: If this value is 1 there is no opening in the lid for the TSOP in the IR board. Change to 0 if you want the opening.
- **Suppress_IR_Learn_Hole**: If this value is 1 there is no opening in the lid for the TSMP in the IR board. Change to 0 if you want the opening.
- **Case_Screw_Hole**: Use M3_Brass_Case_Hole or SM6_20_Case_Hole depending on screw type.
- **CountersinkDist**: Use M3_Sink_Dist or SM6_20_SinkDist depending on screw type.
- **CountersinkAngle**: Use M3_Sink_Angle or SM6_20_SinkAngle depending on screw type.
- **LidHole**: Use M3_Lid_Hole or SM6_20_Lid_Hole depending on screw.
- **DoorGap**: 0.3 mm tolerance for the USB cover door.

There are a number of other parameters that you can try adjusting as well. The names should be reasonably self-explanatory. After making these changes, you will have to export new STL files and print them.
Preparing the Boards

Several of the boards need some preparation before they can be installed into the case. You will need to solder some wires on some of the boards. You can use soft silicone covered stranded wire for most of the connections, however some of them have to be pushed into a socket, so you will need solid core wire. We will make special note of which wires need to be solid core. Any other wires can be solid or stranded.

Preparing the Power Boost

The Power Boost 1000c requires a bit of assembly. You need to solder in the large USB A connector onto the board first. There are four small signal pins and 2 large mounting pins that need to be soldered.

Using wire cutters or side cutters, clip off one of the outside pins of the SPDT slide switch. Bend the other two leads 90° and insert them into the EN and GND holes of the power boost as seen in the image below. The switch should rest on the edge of the board as shown in the end view on the lower right.

You should solder 3 wires to the PowerBoost board. We like to use red wires for power and green wires for ground, but you can use any colors you want. Make a red wire about 3 inches long connected to the USB pin. Make a red wire about 5 inches long connected to the 5Vo pin. Make a green wire about 5 inches long connected to the GND pin that is adjacent to the 5Vo pin. Note that there is another GND pin, however it is already occupied by the slide switch. These wire lengths are approximate and we will trim them to the proper length later. See the image below.

Preparing the TFT Display Board
On the backside of the TFT display, there is a small solder pad labeled LITE. You need to solder a wire from that pad to the pad corresponding to pin 13 which is the fourth pin from the bottom on the 12 pin side of the feather socket. In the image below is the orange wire. You might want to tape down that wire or perhaps glue it to the board after you solder the ends in place. You will also need to connect a green ground wire about 3 inches long to the ground pad which is the fourth from the bottom on the 16 pin side. You will also need to make a 3 inch red power wire soldered to the pad for the USB pin of the Feather. It is the third from the bottom on the 12 pin side. These power and ground wires will come out from underneath the Feather board which will set on top of them. Refer to the image below for the exact locations.

Preparing the Feather M0 BLE

You should solder a standard set of header pins onto the Feather M0 BLE. The pins come with the board. You should also check to see that the Bluefruit firmware has been updated. For complete details on the use of this board check out its learning guide.

- Adafruit Feather M0 Bluefruit LE Learning Guide (https://adafru.it/FMg)

After you have confirmed that it is working using the sample sketches in that learning guide, you should disconnect the USB, plug it into the TFT Display Board, reconnect the USB, make sure that the on-off switch on the display board is on and try running some of the sample sketches for the display which can be found in this tutorial.

- Adafruit 2.4" TFT FeatherWing Learning Guide (https://adafru.it/vvE)
Wiring the AT Jacks

In all of my versions of the "Ultimate Remote, I have controlled it using a series of 3 ultralight touch micro switches connected to the device by a four strand ribbon cable. However the design we are presenting here uses 3 standard 3.5 mm (1/8 inch) mono panel mount audio jacks. A variety of assistive technology (AT) switches are available that can be plugged into these jacks. This is one area where you may want to depart from our design and come up with a connection method of your own. These three jacks are wired into the A0, A1, and A2 Feather pins, however all of the analog pins A0 through A5 may be used as either analog or digital inputs.

Although the images we show here are modeled with individual wires, it might be convenient to use a four wire ribbon cable to connect these three jacks and a common ground wire.

Because these wires plug into the feather socket of the TFT board, they will have to be solid-core or you have to solder on a four pin header that can plug in.

Arrange the three jacks in a row and connect the outer "ring" terminals together using short pieces of wire. Off the end of one of the jacks connect a longer piece of wire about 3 inches long and it must be a solid wire or have a solid end. This wire is shown in dark green on the image below. Then solder on individual wires to the "tip" terminals each about 3 inches long and they must have a solid end. Each of these wires will plug into the A0-A2 terminals on the TFT board. We have illustrated these in different shades of blue.
Mounting and Wiring Parts

Now to begin mounting all of the parts in the case. If you already have the Feather M0 BLE inserted into the TFT Board, you might want to remove it until you get the TFT mounted. Mount the TFT display into the front part of the case using 4 screws and nuts. 2.5 mm screws work best. If you do not have the proper size screws, you could also glue the board in place using hot glue or E6000 glue. Make sure that the USB connector on the Feather board is at the top of the case so that it can be accessed through the area of the door in the case.

After mounting, press the Feather M0 BLE board onto the TFT display. The socket on the TFT display is especially shallow, so the board will not go in all the way. Make sure you have it in sufficiently that all the pins make good contact. See the image below.

Mount the three AT jacks in the side of the case and secure them with the nut that comes with the jacks. Connect the wires from the jacks to the GND and A0, A1, and A2 pins of the TFT board.
Solder a green ground wire about 3” long to the center top terminal of the 2.1 mm barrel jack. Position the jack into its receptacle in the case and secure it with glue as seen in the image below.

Connect the 4400mAh Li-Ion battery to the PowerBoost board using the built-in JST connector. Position the PowerBoost in the end of the case with the component side down. There are two plastic pins which insert into the mounting holes on either side of the micro USB jack. When we put the lid on the case, it will hold the Power Boost securely in place. Secure the battery in position in the case using tape or a very small amount of hot glue. Do not use CA glue or other permanent glues because you will eventually need to replace the battery. It should be very close to the jacks because you will need to leave an empty space on the other side for the T-nut which will extend down from the lid.

Now connect the red USB wire from the PowerBoost to the top rear terminal of the barrel jack. Trim the USB wire to an appropriate length and solder it in place. See the image below.

Position the IR board in the upper left corner of the case. There are two pins in the case that fit into the mounting holes. The yellow wire from the IR board is going to plug into pin 12 just outside the Feather on the TFT display. You should trim it to appropriate length and strip it and plug it into the socket of the TFT. You will recall that we insisted this be a solid wire so that it could plug into the socket.
If you are using the receiver chip you also need to trim the brown +3.3v wire and insert it into the "3v" pin of the TFT near the upper left corner of the Feather board.

Also the white receiver wire should go into pin 11 adjacent to the LED wire that we previously plugged into pin 12.

Now it is time to do the final wiring. We need to tie together red 3 power wires. There is one coming from the "5Vo" pin of the Power Boost on the left and 2 wires one from the TFT board underneath the Feather and the other from the IR board. Slide a piece of heat shrink tubing over the wire on the left, splice the three wires together, and then position heat shrink over the joint and shrink it. For the ground wires there are 4 wires to be spliced together. The two on the left extend from the Power Boost and the 2.1 mm barrel jack. The two on the right come from the TFT board underneath the Feather and from the IR board. Again position a piece of heat shrink tubing over 2 of the wires. Connect with four wires together in a splice and slide the heat shrink into place and shrink it. The image below shows the final wiring.

You may want to do some software testing before putting the lid on the case.

Before securing the lid in place, insert the T-nut into the case. You may need to tighten it into place by inserting a 1/4"-20 screw and washer to pull it tight. See the assembly video below for more information. This will provide a 1/4"-20 mounting receptacle to mount the entire unit on a camera mount or other type of bracket using this standard mounting technique.
Here is what the case looks like fully assembled with either M3 or #6-20 screws. After you have programmed the software, you can slide in the sliding door covering the USB port of the Feather. Finally optionally you can glue in place a protective shield over the LEDs to protect them from getting bumped around.
Building the Receiver
Dongle

If you’re going to use the Ultimate Remote to send keyboard and mouse commands to a computer, you will need a receiver device. It requires a TSOP38238 or TSOP38438 receiver chip and a Trinket M0 board. Bend the leads on the TSOP so that the pins will go into the RX pin (pin 3), the ground pin, and the 3v pin on the Trinket as shown in the image below.

Solder the leads in place and trim the excess. It will snap fit into the 3D printed case. There are two pins in the case that lineup with the mounting holes on the left end of the board. The lid will snap fit into place.
Here is a YouTube video that describes the entire process.
Installing Libraries and the Sketch Software

Library Installation

There are number of Arduino software libraries that must be installed for this project. You should configure your Arduino IDE to operate Feather M0 BLE and Trinket M0 boards. You also need to install a number of libraries for the TFT Display board. See the learning guides for those particular boards for complete details on what software is necessary to operate those boards. As suggested earlier, you should run some of the Bluefruit demo sketches as well as graphics test sketches as described in those tutorials.

All of the IR functions for this project depend on my IRLib2 libraries which are available on GitHub at this link:

https://adafru.it/vwF

That library cannot be installed using the Library Manager. You have to install manually. Installation of the IRLib2 library is as follows:

1. Visit the IRLib2 page on GitHib (https://adafru.it/vwF).
2. Select the “Download ZIP” button, or simply click this link (https://adafru.it/vxa) to download directly.
3. Uncompress the ZIP file after it’s finished downloading.
4. The resulting folder should be named "IRLib2-master" and will contain 5 separate folders. That is because IRLib2.x is actually a collection of 5 libraries that work together. Sometimes in Windows you’ll get an intermediate-level folder and need to move things around.
5. Copy all five folders into your Arduino library folder alongside your other Arduino libraries, typically in your (home folder)/Documents/Arduino/Libraries folder. Libraries should not be installed alongside the Arduino application itself.
6. Restart the Arduino IDE if it’s currently running

This repository consists of a total of five libraries each of which must be in your arduino/libraries/ folder. So for example it should be installed as follows...

- arduino/libraries/IRLib2
- arduino/libraries/IRLibFreq
- arduino/libraries/IRLibProtocols
- arduino/libraries/IRLibRecv
- arduino/libraries/IRLibRecvPCI

Do not install them in a single folder such as this...

- arduino/libraries/IRLib2_master
  - IRLib2
  - IRLibFreq
  - IRLibProtocols
  - IRLibRecv
  - IRLibRecvPCI

Here is a tutorial (https://adafru.it/m3e) that walks through the process of manually installing libraries that are not available through the Arduino Library Manager.
There is a minor modification that you need to make to IRLib in order to get it to work with this project. The default output pin for an Adafruit Feather M0 is pin 9. However that particular pin is used to control the TFT Display. So we have to change the default to pin 12 which is how we wired this device.

Find the file `Arduino/libraries/IRLibProtocols/IRLibSAMD21.h`. This file contains the defaults for using any board that uses the SAMD21 processor. At approximately line 91 you will find the following code:

```c
#define IR_SEND_PWM_PIN 9
```

You must change the last line to read:

```c
#define IR_SEND_PWM_PIN 12
```

**Warning:** If you are using IRLib for other projects that expect the default to be pin 9, you will have to edit this line again to put it back to its normal default. Unfortunately there is no way to specify output pin number in your end-user sketch. You must edit the library.

**Sketch Installation**

There are three sketches in the repository that you downloaded for this project. The one called `ultimate_remote` should be uploaded to your Feather M0 BLE board. The one called `trinket_dongle` should be uploaded to the Trinket dongle obviously. The software has already been configured to work with my Samsung 3d TV, my Spectrum Cable DVR, my Sony amplifier surround sound system, and a Samsung 3d Blu-ray player. The chance that you have the same model using the same protocol as I do is pretty slim. But there is one part of the remote that you can test immediately and that is the mouse and keyboard functions. See the section “Mouse and Keyboard Pages” on the page [Operating the Device](https://adafru.it/FMh) and give it a try.

You will need to reconfigure the software to work with your particular TV, DVR or other consumer electronic device. We suggest you play around with the mouse and keyboard features first because it does not need configuration or customization. If you did not build a dongle and do not plan use your device for mouse and keyboard emulation, then you can still learn to navigate the menus, but of course it won't do you much good until you can configure it for your own devices. After playing around a bit, you should proceed to section [Configuring for Your IR Devices](https://adafru.it/FMi).
Operating the Device

Even though we've not yet customized the remote for the particular devices you want to control, we need to explain how the remote works so that you can understand what it is being configured. Also note that you can use the mouse and keyboard emulation portions of the device without any customization, so you can start using that portion of the device right away.

Here is a YouTube video that demonstrates many of the functions of the remote. You can read more about it in detail in the sections below.

Using AT Switches

The device is designed to operate using 3 standard AT switches which are connected through the three jacks on the side of the case. We will refer to them as LEFT, RIGHT, and SELECT. They are used to move the cursor around the display screen. We've also implemented a system of "cording" which means pressing more than one button simultaneously to get a different function. If you push the LEFT and SELECT buttons simultaneously, the cursor will move up and if you push the RIGHT and SELECT buttons simultaneously the cursor moves down.

We should note that just because we've built the device to use ordinary switches, with a little extra programming, other input methods such as sip-and-puff, flex sensors, force resistors, and capacitive touch could be used to control the device. That is beyond the scope of this tutorial. Contact us at ATMakers.org on our Facebook group here (https://adafru.it/FMj) for more information and assistance.

Using the Touchscreen

Additionally, we have taken advantage of the fact that the TFT display has a resistive touchscreen over it. Although you probably would not use the touchscreen for everyday purposes, it is available for testing and demonstration purposes without the need to connect switches. If the disabled person needed assistance from an attendant or caregiver, they could operate the device using the touchscreen rather than the user's usual switches. If the touchscreen is a bad idea for your day-to-day use, the program can be recompiled easily to disable it.

The touchscreen is divided into five sections for up, down, left, right, and select as seen in the drawing below. Note that there's nothing on the screen to indicate where these areas are, the select section is the center third of the screen and the other directions surround it.

Serial Monitor Debugging

Near the top of the ultimate_remote sketch on the front page is a definition
#define MY_DEBUG 0

Changing this value to a 1 and recompiling turns on serial monitor debugging. It also allows you to navigate the screen by typing commands into the serial monitor. You can type in "u", "d", "l", "r", "s" for up, down, left, right, and select.

Navigating the Screen

When operating the device, think of the characters on the screen as a menu and you are going to navigate around it and select the function that you want to transmit.

Using the AT switches, you can move the cursor left or right and then select the function that you want. As previously mentioned, if you press LEFT and SELECT at the same time, the cursor will move up and if you press RIGHT and SELECT at the same time the cursor will move down. However some users will not be able to use this multi-button system so there is an alternative.

As you move the cursor past the right edge of the commands, you will see a down arrow appear. If you press SELECT while the down arrow is visible, the cursor will move down one row. If you press RIGHT again when the down arrow is available, an up arrow will appear and pressing SELECT will cause the cursor to go up. If you continue to press RIGHT, the up or down arrow will disappear and the cursor will wrap around to the beginning of the same row.

Similarly as you go up or down off the top or bottom of the screen, the cursor wraps around.

This wraparound feature both horizontal and vertical means that technically you really only need 2 buttons. A direction button (either RIGHT or LEFT) and a SELECT button. It takes longer to navigate the screen if only one direction button is used, but it is possible to completely operate the device with just two buttons. At various times late in the day when my fingers are cramping, I often operate the device using only 2 buttons. I've also been in situations where my RIGHT or LEFT button failed due to a loose wire. As long as my SELECT button and at least one other direction button works, I was still able to use the device until I could get it repaired.

Menu Pages

The menus are organized into pages of commands. Each page can consist of up to 9 rows of 14 columns of commands. (Note: most of these demonstration pages are only 10 columns wide because they were designed for a version of this remote that used the display screen in a portrait orientation instead of landscape.)

The top row of the screen is used to select which page you want. The top row of symbols never changes. However as you select different pages, the remaining rows will display different sets of commands. You can create your own pages and organize them however you want. We are going to describe the pages that I use. This will serve as a demonstration of how the system works. Across the top row you will see the letters "CMKATBX" followed by a Bluetooth symbol. Those characters stand for "Cable, Mouse, Keyboard, Amplifier, TV, Blu-ray, eXtra". The default page when the system turns on is my cable page so that is what you are looking at. The remaining lines on the screen are all commands to control my cable DVR.
Your cable or satellite box or DVR may or may not have all of these functions and it may have additional functions. You will be able to design your own page and commands. We will explain this cable page in some detail so that you can see what is possible. We will not go through a detailed explanation of the commands for my surround sound amplifier, TV commands, or Blu-ray player. The eXtra page contains commands from the Adafruit Mini Remote Control which I use on many of my maker projects and to demonstrate IR functions.

Selecting the Bluetooth symbol at the end of the top row changes the remote from infrared control mode to Bluetooth mode so that you can use the device for iOS switch control on iPhone or iPad devices. The Bluetooth switch control mode is described in a separate section of this tutorial.

Let's look in more detail at the first page with the cable commands. We have designed some custom symbols to represent the typical commands you would find in a TV or cable remote control. For more information on these symbols see the tutorial linked here (https://adafru.it/FMk).

The first row of commands perform the following functions on my DVR: Jump back, Rewind, Play, Fast-Forward, Pause, Stop, Live, Mute, Volume Up, and Volume Down.

In the next row we used a mixture of custom symbols and regular characters for the functions. They are: Channel Up, Channel Down, Info, Favorites, preVious Channel, eXit, Guide, List, Menu, and Record. When you select Record, a pop up message will appear on the screen asking "Are You Sure?" You have to press the RIGHT button to confirm that you really want to record. This has saved me from accidentally recording a show I didn't want and causing my DVR to overflow.

The third row of commands are the left, right, up, down, select, page up, page down, A, B, and C buttons from my DVR remote. Those arrows are used to navigate the on-screen guide and the list of recorded programs as well as the cable box's menu system. The A, B, and C buttons perform miscellaneous functions on the DVR.

The next row consists of PIP (picture-in-picture) functions and some others. They are Toggle PIP, Move PIP, Swap PIP, PIP Channel Up, PIP Channel Down. This is followed by a special command for navigating the on-screen guide that goes to the Previous Day, or Next Day. The final three commands on this row are "CTQ" which mean Cable Power, TV Power, and Quit. The quit command doesn't transmit anything via IR. It simply blanks the display and it locks out the switches for 10 seconds. I select this when taking the remote off or on my wheelchair or removing the buttons from my...
Note also that after five minutes of inactivity, the screen will blank itself. It actually turns off the backlight thus saving battery power. The next time you press a button, the screen will reappear however when the screen is blank, the SELECT button is disabled so you don't accidentally select a command that you can't see. If you want to completely disable the device you can turn it off with the power switch at the bottom of the case.

Mouse and Keyboard Pages

Let's look in detail at the Mouse and Keyboard pages of commands. IRLib contains a special protocol of my own design for doing mouse and keyboard control. The dongle is programmed to receive these special commands and perform the actual functions on your computer.

You should navigate to the top row and select the "M" to switch to the mouse page.

The first row is basic mouse movement in any of eight directions horizontally, vertically or diagonally. The "l" in the center of the row does a single left click of the mouse. The "r" at the right end of the row does a single right click. You can select these rapidly for double clicks.

The next row begins with "LRM". These are toggle buttons for the Left, Right, and Middle mouse buttons. If you click them once it presses and holds the mouse button until you press it again. The "wW" commands move the mouse scroll wheel up or down. The next three characters "SCA" are keyboard toggles for Shift, Control, and Alt. This allows you to do things like "Shift Click" or "Control Click". Like the mouse toggles, one press turns them on and the next press turns them off. The final "fs" changes the mouse speed faster or slower. The default is 10 units per command.

Note: the receiver dongle is programmed to release all mouse and keyboard toggles of any kind after two minutes of inactivity. That way if you accidentally leave the shift key on for example and go away to do something else and forget how you left it, you can be assured it will always revert back to normal. It also resets the mouse speed to the default 10.

Although this page is primarily a mouse page, it also includes several other handy keyboard commands. The third row
begins with keyboard presses of Home, Up Arrow, Page Up, Left Arrow, Enter, Right Arrow, End, Page Down, and Windows Key. (Although we have not tested this on a Mac computer it should work as a command key on those systems.)

The fourth row consists of several other useful characters. The "zyxvc" commands are all control characters for Ctrl-Z, Ctrl-Y etc. On most programs these control characters do "Undo", "Redo", "Cut", "Copy", and "Paste". That "E" is Escape. The remaining four symbols are "Backspace", "Insert", "Delete", and "Tab". The fifth row contains a space bar followed by a "0". That zero means reset all toggles. This includes all three mouse buttons as well as Shift, Control, and Alt toggles. It also resets the mouse speed to the default.

You should now navigate to the "K" at the top of the page to select a full keyboard page.

This simulates a full QWERTY keyboard including 12 function keys. The top row begins with a backwards accent or apostrophe character followed by 1 through 0, -, =, and finally Backspace. The second row is the standard first row of the keyboard. The symbol at the end is Delete. The third row is the typical middle row of the keyboard with an Enter followed by the Space Bar at the end. It seemed to be more convenient in that location then putting it at the bottom of the keyboard like a physical keyboard would. The fourth row covers z through "/" followed by toggles for Shift, Control, and Alt. These work just like the keyboard toggles in the mouse screen. One press turns them on. A second press turns them off. Note that when you choose the "S" Shift toggle the display changes to display uppercase characters and to replace the numbers and punctuation marks with their typical shifted versions on a physical keyboard. Here is how the screen looks after selecting the Shift.
The next row consists of navigation keys Home through Page Down as duplicated from the mouse page. The "wW" are Windows keys with the small "w" giving a momentary press and the "W" is a toggle so that you can use Windows shortcuts such as "Win-A" to bring up the notification area. There are a number of these such shortcuts built into Windows. The "e" is the escape key and the "0" is the toggle reset which releases all Shift, Control, and Alt toggles. The final row are the function keys F1-F12.

Without going into a lot of detail, here's a brief overview of the other pages. As mentioned previously the "A" page is for my surround sound amplifier. It looks like this.
The "T" is for my TV controls apart from the usual power and volume controls which are on the cable page where they are more handy.

The "B" page is for my Samsung 3D Blu-ray player. Note that it has 2 sets of volume controls. The one in the upper right is for the TV and the one in the lower right is on my surroundsound amplifier because sometimes I watch Blu-ray with or without the surround sound on.

The "X" page is extra commands that I had programmed to emulate the Adafruit Mini-Remote. I arranged it in a narrow format of three columns so that looks just like the remote itself. The software has a little difficulty handling a narrow column and I haven't bothered to completely debug it.

Bluetooth Switch Control

Apple iOS operating systems have an accessibility feature called "Switch Control". It allows you to control an iPhone or iPad without using the touchscreen. You need to connect a device that emulates a standard Bluetooth keyboard. By pressing different keys on this virtual keyboard, you can move a cursor around the screen of your iOS device and
select programs, type, drag-and-drop and basically do anything you could do with the device using its traditional touchscreen controls.

In a separate tutorial here, we have described in detail how this feature works and how to configure your iPhone or iPad to use it.

- Using Bluefruit BLE to Give Disabled Users Switch Control Access to iOS Devices(https://adafru.it/FM4)

We have program software to send "L" when you push the LEFT switch, "R" when you push the RIGHT switch, and "S" when you press the SELECT switch. The software also allows you to use long keypresses so that a long press of one of the switches can have a different effect on the iOS device. This is all explained in the switch control tutorial.

There are three ways to exit switch control mode. If you push all three switches simultaneously it will exit. If you push 2 switches for one half second it will exit. If you press 1 switch for more than 8 seconds it will exit. After exiting switch control mode, the remote will return to whichever page you were on before you switch into Bluetooth switch control.

Charging the Device

You can recharge the battery using the micro USB port of the power boost at the bottom of the device or by using the 2.1 mm barrel jack. Although the device will operate if powered through the micro USB port of the Feather M0 BLE, it will not recharge the device when connected through that port.

Warning: only use a charger on the barrel jack that outputs 5 volts. See the specifications on the power boost for details of what it can handle.

We recommend this charger for recharging the device using the barrel jack.

5V 2A (2000mA) switching power supply - UL Listed

$7.95
IN STOCK
Add To Cart
Configuring Your IR Devices

We wish we could tell you that adding new functions to this remote was as simple as pointing another remote at it and have it learn the code directly. Unfortunately, we are not yet at that level of sophistication. The process is a bit more complicated. Here is an overview of the process:

1. You will load a special decoding sketch into the Trinket Dongle or, if you did not create a dongle, you can load it into the Ultimate Remote itself since you should have installed TSOP receiver device on the IR board.
2. You will then point the remote for your TV or other device at the receiver and press a button. It will try to detect which of the 11 protocols that IRLib supports. The result will be a print out on the serial monitor of the protocol type and a hexadecimal number of to 32 bits long that is the code for that particular button.
3. You will then copy that information into the `ultimate_remote` sketch and tell it what symbol you want to be displayed.

Next, we will now walk you through these steps in more detail. To illustrate this process we are going to add a new page of commands for an RCA TV. Of course, unless you happen to have this particular type of TV, doing the steps exactly like we do them will not do you much good. But you should follow along and adapt this method to your own device. Substitute in filenames and button names that make sense for your TV or whatever it is you want to control. The steps will be the same. The details will be unique to you.
Recording Your IR Signals

Adding a New Code Definition File

The image above is the remote control for my new roommate's RCA TV. We are going to capture the IR codes for all of the buttons on that remote and then design a new page of commands so that we can control that device with our Ultimate Remote.

Also open the ultimate_remote sketch and do a File -> Save as with a new name something like ultimate_remote_with_rca. Take a look across the top of the Arduino IDE and you will see a number of tabs for multiple files that are part of this program. The files that are named (whatever)_codes.h contain the codes for all of the commands that are used by the device. We are going to create a new tab called rca_tv_codes.h that will contain the codes for the RCA TV.

In the upper right corner of the Arduino IDE is a small downward pointing triangle. Click on that triangle and there will be a drop-down menu with a variety of options. You can use that triangle to navigate to existing tabs. We want to use the "New Tab" option.
At the bottom of the IDE a dialog will open where you can enter the name of the new tab page that you want to create. We will call it `rca_tv_codes.h`. Obviously you will want to pick a filename that makes sense for your device. Click on the "OK" and a blank tab will appear.

In the blank page that you just created start typing a define command for every button on your remote. Give it a unique name that identifies the device and the function. For example our begins with the following few lines and continues with descriptions for all of the buttons.

```c
#define RCA_UP
#define RCA_LEFT
#define RCA_OKAY
#define RCA_RIGHT
#define RCA_DOWN
```

We recommend that you make an entry for all of the buttons whether or not you think you might use them or not. It makes it easier to capture all of the codes while you have everything set up to do so. Later you might want to edit your command page and add or remove some codes and you don't want to have to set everything up again to capture more codes.

Note: You will probably not be able to enter number keys using the Ultimate Remote software as currently written. That is because when entering numbers on a TV or DVR you typically have to enter them quickly after one another. You would not be able to navigate the Ultimate Remote Screen in time. We suggest you still capture the codes because future versions of the software will be able to queue up several digits and then send them in sequence.

Capturing IR Codes

You should load the sketch titled `dongle_dump` in the Arduino IDE and upload it to the Trinket dongle. (Or if you did not make a dongle, upload the sketch `remote_dump` to the Ultimate Remote itself.) Compile and upload the sketch and open the serial monitor.

Point the remote control for your TV or other device at the receiver dongle or the receiver on your Ultimate Remote. After we pressed the "Up Arrow" button on our remote we saw the following on the serial monitor:

```
Ready to receive IR signals
Decoded NEC(1): Value:C738B847 Adrs:0 (32 bits)
```

This says that we were able to successfully decode NEC protocol which is protocol number 1. The value received was "C738B847" and the Address was 0. "Address" is an additional bit of data used by some protocols. In this case it is
unused. Finally, we are told this is a 32-bit protocol. In fact, all NEC protocols are 32 bits, but some protocols such as SONY have a variable number of bits.

Can we now go back to the page `rca_tv_codes.h` and enter that information into the define statement for the Up Arrow. Note that because this is a hexadecimal number you must precede it with `0x`. We repeated this process for all of the buttons on the remote. Here is what the entire set of entries will look like after we have edited in the codes.

```
#define RCA_UP 0xc738b847
#define RCA_LEFT 0xc7387887
#define RCA_OKAY 0xc738d02f
#define RCA_RIGHT 0xc73838c7
#define RCA_DOWN 0xc738c837

#define RCA_POWER 0xc73840bf
#define RCA_HOME 0xc738da25
#define RCA_EXIT 0xc738a758
#define RCA_BACK 0xc73828d7
#define RCA_APPS 0xc738ea15
#define RCA_NETFLIX 0xc738d02f
#define RCA_VUDU 0xc738c738
#define RCA_BLANK_LEFT 0xc73847bf
#define RCA_BLANK_CENTER 0xc73842bd
#define RCA_INPUT 0xc73858a7

#define RCA_VOLUP 0xc73832cd
#define RCA_MENU 0xc7389a65
#define RCA_CH_UP 0xc738a955
#define RCA_VOLDN 0xc7382ad5
#define RCA_MUTE 0xc7386a95
#define RCA_CH_DN 0xc738728d

#define RCA_1 0xc73850af
#define RCA_2 0xc738906f
#define RCA_3 0xc73810ef
#define RCA_4 0xc73840bf
#define RCA_5 0xc7388877
#define RCA_6 0xc738087f
#define RCA_7 0xc7388877
#define RCA_8 0xc738b24d
#define RCA_9 0xc738c03f
#define RCA_MINUS 0xc7389867
#define RCA_REWIND 0xc738b24d
#define RCA_PLAY_PAUSE 0xc73812ed
#define RCA_STOP 0xc73800ff
#define RCA_FF 0xc738af5
#define RCA_RED 0xc7380ff
#define RCA_GREEN 0xc7389ff
#define RCA_YELLOW 0xc7380ff
#define RCA_BLUE 0xc7380ff
```

When you have completed capturing all of the codes, you should save that file by pressing "Ctrl-S" on your keyboard. See the section on Troubleshooting in this tutorial if you are not able to successfully decode the signal from your remote.
Now that we have captured all of the IR signals and recorded them, we will show you how to create a new page of commands using the signals.
Adding a New Page of Commands

Defining a New Page

The pages of commands that are displayed on the screen of the Ultimate Remote are defined in a file called `my_pages.h`. Open that particular tab. Near the top of the page you will see a series of `#include` statements that look like this.

```plaintext
// These files contain the IR codes for the various devices we want to use.
#include "sa_cable_codes.h"
#include "samsung_necx_codes.h"
#include "sony_amp_codes.h"
#include "samsung_bluray_codes.h"
#include "adafruit_mini_codes.h"
```

Notice that in the Arduino IDE, there is a tab for each of these files. We have already added a tab for the RCA TV codes but that doesn't inform the compiler where and how to make use of that information. We need to edit our list of `#include` statements to add our file. At the bottom of that list enter the following:

```plaintext
#include "rca_tv_codes.h"
```

Just below that section of code you will find the following list of include statements.

```plaintext
// IRLib2 includes.
// Start out with the base sending library followed by the individual protocols we will use and then
// tie everything together with the combo file.
#include <IRLibSendBase.h>
#include <IRLib_P01_NEC.h>
#include <IRLib_P02_Sony.h>
#include <IRLib_P05_Panasonic_Old.h>
#include <IRLib_P07_NECx.h>
#include <IRLib_P08_Samsung36.h>
#include <IRLib_P12_CYKM.h>
#include <IRLibCombo.h>     // After all protocols, include this
```

These include files are part of IRLib2 and they contain the code that tells your device how to send signals for the particular protocol. Our RCA TV uses NEC protocol and as you can see we've already included `IRLib_P01_NEC.h` so we don't need to make any changes here. But if you're particular device used a different protocol, perhaps protocol 3, which is called RC5, you would also have to add the following statement to the list.

```plaintext
#include <IRLib_P03_RC5.h>
```

Warning: You MUST put these include statements in numerical order by their protocol number and conclude with `IRLibCombo.h` or IRLib2 will not compile or operate properly. See the documentation for IRLib2 for further information.

These individual protocol files are stored in `arduino/libraries/IRLibProtocols`. 
You should now continue to scroll down the `my_pages.h` file. The next thing you will find is a series of definitions that make some of the other coding in this file a little easier. For now you can ignore that section.

It is followed by a series of defines that enumerate the pages. It looks like this:

```c
// These are names of the pages
#define PAGE_CBL 0
#define PAGE_MSAR 1
#define PAGE_KB 2
#define PAGE_AMP 3
#define PAGE_TV 4
#define PAGE_BLU 5
#define PAGE_TEST 6
// The BLE page isn’t really a page. It shifts the device into Bluetooth switch control mode.
// It has to be the last page in the list for everything to work right.
#define PAGE_BLE 7
```

We will want to add a `#define PAGE_RCA_TV`. You can order the pages in any order you want, but the `PAGE_BLE` absolutely must be the highest numbered page. So we will make our new page 7 and change `PAGE_BLE` to be 8 as follows:

```c
#define PAGE_TEST 6
#define PAGE_RCA_TV 7
// The BLE page isn’t really a page. It shifts the device into Bluetooth switch control mode.
// It has to be the last page in the list for everything to work right.
#define PAGE_BLE 8
```

The next section of code reads as follows:

```c
// Each page of commands can have a variable width up to COL_MAX. Define the widths here.
const uint8_t Page_Widths[PAGE_BLE+1] = {10, 10, 14, 10, 10, 10, 3};
```

It defines the width of each of your pages. Most of our pages are 10 commands wide because we developed this on a version of the Ultimate Remote that would only do 10 columns wide. Anything too wide can make it more difficult to navigate especially if you are only using 2 instead of 3 buttons to control the remote. We are going to make this new page 10 columns wide. Note that the third page is a keyboard page which is 14 columns wide. The final page is the "Extra" page that is only 3 columns wide. We have added our page after that extra test page so we will modify this array to say that our new page is 10 columns wide by editing it as follows:

```c
const uint8_t Page_Widths[PAGE_BLE+1] = {10, 10, 14, 10, 10, 10, 3, 10};
```

Now that we have set up the new page, we will show you how to define the rows of commands for that page in the next section of this tutorial.
Defining New Rows

Understanding Commands

Continuing downward on `my_pages.h` the next section is a large two dimensional array containing the definitions of the rows of commands. Each entry in the array is a structure of type `cmnd_t` which is defined in the main tab of the program as follows.

```c
typedef struct cmnd {
    uint8_t Glyph;    //The character to be displayed
    int32_t Value;    //string of bits to be transmitted or page number two change to
                     // if Special == SPL_CHANGE_PAGE
    uint8_t Protocol; //Sony, NEC etc.
    uint8_t Special;  //special handling flag or number of bits
} cmnd_t;
```

If you're not a C++ programmer, this will look like gibberish to you. Don't worry, we will explain what it means. Let's look at one of the existing rows of commands back in `my_pages.h`. At approximately line number 129 you will see the following code. This is the second row of my definitions for my Samsung 3D TV.

```c
#define ROW_TV_2 13
{{'m',TV_MENU,NECX,0}, {'t',TV_TOOLS,NECX,0}, {MY_LEFT_ARROW,TV_LEFT,NECX,0},
 {MY_RIGHT_ARROW,TV_RIGHT,NECX,0}, {MY_UP_ARROW,TV_UP,NECX,0}, {MY_DOWN_ARROW,TV_DOWN,NECX,0},
 {'*',TV_SELECT,NECX,0}, {'x',TV_EXIT,NECX,0}, {'r',TV_RETURN,NECX,0}, {'3',TV_3D,NECX,0}},
```

The first line is just a comment reminding us what commands are going to be on this particular row of the page. The next define is a definition of the row name and number. Each row needs a unique name and number. The row number needs to be in order with the definitions above and below it. This numbering system applies to the entire remo, not just the page we are working on.

The next three lines are the series of 10 commands that make up the row. The entire row is enclosed in scroll braces {} followed by a comma. Each of the commands within the row consists of 4 items separated by commas and enclosed in scroll brackets. Let's look at the first command it reads `{'m',TV_MENU, NECX,0}`

This tells us that the character to be displayed on the screen is the letter 'm'. The IR code to be transmitted is named TV_MENU. That value is defined in the file `samsung_necx_codes.h` where I have all of the IR codes for this particular TV. The next item is the protocol number. In this case it is NECX. The possible values for this protocol number are defined in `Arduino/Libraries/IRLibProtocols/IRLibProtocols.h` as follows. For more information see the IRLib documentation.
The final item is a special flag that has multiple purposes. We will describe later. In this instance it is simply 0.

The next entry is to bring up the Tools menu on the TV. Its display letter is 't'. The next 4 entries are for the arrow keys on the remote. Rather than use a text character for the display we are going to use a special graphics character called MY_LEFT_ARROW as well as special characters for right, up, and down. You can see a complete list of the available special graphics characters in the file SymbolMono18pt7b.h.

Defining New Commands

Now that we've seen how the commands are laid out, let's define new commands for the RCA TV. We are going to create a page with three rows of commands. We are not going to define a button for absolutely every command on the remote because we don't use everything very often. Also as mentioned earlier, it isn't really practical to enter channel numbers with the Ultimate Remote at this time.

In my_pages.h scroll down to approximately line number 210. You will see the end of the keyboard page definitions ending with ROW_KB_6 32. Below that are some empty commands we've thrown in just in case we accidentally get some index out of range. We are now going to insert three rows of commands after the end of the keyboard definitions but before these extra entries. It will look like this.

```c
#define UNKNOWN 0
#define NEC 1
#define SONY 2
#define RC5 3
#define RC6 4
#define PANASONIC_OLD 5
#define JVC 6
#define NECX 7
#define SAMSUNG36 8
#define GICABLE 9
#define DIRECTV 10
#define RCMM 11
#define CYKM 12

#define ROW_RCATV_1 33
{{MY_POWER_SYMBOL,RCA_POWER,NEC,0}, {'i',RCA_INPUT,NEC,0}, {MY_CH_UP,RCA_CH_UP,NEC,0},
{MY_CH_DOWN,RCA_CH_DN,NEC,0}, {'+',RCA_VOLUP,NEC,0}, {'-',RCA_VOLDN,NEC,0},
{'0',RCA_MUTE,NEC,0}, {'?',RCA_EPG_I,NEC,0}, {'b',RCA_BACK,NEC,0}, {'-',RCA_MINUS,NEC,0}},

#define ROW_RCATV_2 34
{{'m',RCA_MENU,NEC,0}, {MY_LEFT_ARROW,RCA_LEFT,NEC,0},{MY_RIGHT_ARROW,RCA_RIGHT,NEC,0},
{MY_UP_ARROW,RCA_UP,NEC,0}, {MY_DOWN_ARROW,RCA_DOWN,NEC,0}, {'*',RCA_OKAY,NEC,0},
{'x',RCA_EXIT,NEC,0}, {'h',RCA_HOME,NEC,0}, {'n',RCA_NETFLIX,NEC,0}, {'v',RCA_VUDU,NEC,0}},

#define ROW_RCATV_3 35
{{MY_REWIND,RCA_REWIND,NEC,0},{MY_PLAY,RCA_PLAY_PAUSE,NEC,0},{MY_FF,RCA_FF,NEC,0},
{MY_STOP,RCA_STOP,NEC,0},{'r',RCA_RED,NEC,0}, {'g',RCA_GREEN,NEC,0},
{'y',RCA_YELLOW,NEC,0},{'b',RCA_BLUE,NEC,0},{'a',RCA_APPS,NEC,0}, {' ',0x0,0,0}},
```
The first two rows have 10 commands each in the final row has only 9. We replaced it with a blank entry because we didn't need it. As you can see we use a combination of special symbols such as `MY_POWER_SYMBOL` and regular print characters such as 'i' for the input key. The IR codes such as `RCA_POWER` came from the file `rca_tv_codes.h` that we created earlier.

There are just a couple of more items we need to edit to add these rows and the new page that we've created. Just below the command definitions you will see the following code:

```c
const char Num_Rows_in_Page[] = {5, 6, 7, 4, 5, 8, 1};

// Define which pages contain which rows
const uint8_t Pages[][10] = {
  /*Cable*/   {ROW_PAGES, ROW_CBL_PLAY, ROW_CBL_CH, ROW_CBL_ARROW, ROW_CBL_PIP},
  /*Mouse*/   {ROW_PAGES, ROW_MSAR_1, ROW_MSAR_2, ROW_MSAR_3, ROW_MSAR_4, ROW_MSAR_5},
  /*Keyboard*/{ROW_PAGES, ROW_KB_1, ROW_KB_2, ROW_KB_3, ROW_KB_4, ROW_KB_5, ROW_KB_6},
  /*Amp*/     {ROW_PAGES, ROW_AMP_1, ROW_AMP_2, ROW_AMP_3},
  /*TV*/      {ROW_PAGES, ROW_TV_1, ROW_TV_2, ROW_TV_3},
  /*Blu-ray*/  {ROW_PAGES, ROW_BLU_1, ROW_BLU_2, ROW_BLU_3, ROW_BLU_4},
  /*Test*/    {ROW_PAGES, ROW_TEST_1, ROW_TEST_2, ROW_TEST_3, ROW_TEST_4, ROW_TEST_5, ROW_TEST_6, ROW_TEST_7},
  /*RCA TV*/  {ROW_PAGES, ROW_RCATV_1, ROW_RCATV_2, ROW_RCATV_3}
};
```

The top row of numbers is just what it says: it is the number of rows in each page. Note that this includes the page selection row at the top. Our new page consists of 3 rows plus the page selection row means 4 rows. The "1" at the end of the list is for the Bluetooth page. So we will insert "4" just before that and after the "8" rows of the test page as follows.

```c
const char Num_Rows_in_Page[] = {5, 6, 7, 4, 5, 8, 4, 1};
```

Now we need to tell it which rows go on this page at the bottom just after the section which defines the test page edit it as shown below. Note that each of these rows has to be separated by a comma so be sure to add a comma at the end of the Test row.

```c
/* Test*/    {ROW_PAGES, ROW_TEST_1, ROW_TEST_2, ROW_TEST_3, ROW_TEST_4, ROW_TEST_5, ROW_TEST_6, ROW_TEST_7},
/*RCA TV*/   {ROW_PAGES, ROW_RCATV_1, ROW_RCATV_2, ROW_RCATV_3}
};
```

We have now fully defined our new page of commands for the RCA TV but we can't get there from here. We need a way to navigate to that page. Go back up the top of `my_pages.h` at approximately line number 59. You will see the very first row of commands that appears at the top of every screen. The code looks like this:

```c
// An array of rows of commands each row is at most COL_MAX long but could be shorter.
const cmd_t MyCommands[][COL_MAX] = {
  // Page change commands: cable, mouse/arrows, keyboard, kitchen, amplifier, TV, Blu-ray, extra, Bluetooth
#define ROW_PAGES  0
  {'C', PAGE_CBL, 0, SPL_CHANGE_PAGE}, {'M', PAGE_MSAR, 0, SPL_CHANGE_PAGE}, {'K', PAGE_KB, 0, SPL_CHANGE_PAGE},
  {'A', PAGE_AMP, 0, SPL_CHANGE_PAGE}, {'T', PAGE_TV, 0, SPL_CHANGE_PAGE}, {'B', PAGE_BLU, 0, SPL_CHANGE_PAGE},
  {'X', PAGE_TEST, 0, SPL_CHANGE_PAGE}, {MY_BLUETOOTH, PAGE_BLE, 0, SPL_CHANGE_PAGE},
  {',0x0,0,0}, {',0x0,0,0}, {',0x0,0,0}, {',0x0,0,0}, {',0x0,0,0}, {',0x0,0,0}, {',0x0,0,0}},
```

© Adafruit Industries  https://learn.adafruit.com/building-the-assistive-technology-ultimate-remote  Page 44 of 50
Like other rows of commands, each entry consists of 4 parts. The first is the character or symbol that will be displayed on the screen. Next is usually the IR command but these particular commands don't send any IR signals. Instead this field is used to specify the number of the page. We previously defined `PAGE_RCA_TV` at approximately line number 48 of this file. The third entry usually defines the IR protocol but again we are not sending any IR this time so it remains 0. The final item in each command is a flag that we have not previously used. It's the way we tell the program that this is a special different command than the usual IR transmission command. We put in the entry `SPL_CHANGE_PAGE`. There are a variety of these special flags which we will discuss later. After editing this row of commands to include a link to our new RCA TV page it will look like this.

```c
//Page change commands: cable, mouse/arrows, keyboard, kitchen, amplifier, TV, Blu-ray, extra, RCA TV,
#define ROW_PAGES 0
{{'C', PAGE_CBL, 0, SPL_CHANGE_PAGE}, {'M', PAGE_MSAR, 0, SPL_CHANGE_PAGE}, {'K', PAGE_KB, 0, SPL_CHANGE_PAGE},
{'A', PAGE_AMP, 0, SPL_CHANGE_PAGE}, {'T', PAGE_TV, 0, SPL_CHANGE_PAGE}, {'B', PAGE_BLU, 0, SPL_CHANGE_PAGE},
{'X', PAGE_TEST, 0, SPL_CHANGE_PAGE}, {'R', PAGE_RCA, 0, SPL_CHANGE_PAGE}, {MY_BLUETOOTH, PAGE_BLE, 0, SPL_CHAN
{' ', 0x0, 0, 0}, {' ', 0x0, 0, 0}, {' ', 0x0, 0, 0}, {' ', 0x0, 0, 0}},
```

You can now compile and upload the sketch. If you navigate to the RCA page which is designated by "R" you should see in the following page of commands.
Additional Configuration

Editing Pages

We've shown you how to add an entire new page with new rows of commands. It might be easier to edit an existing page of commands. For example if you have a cable or satellite box or DVR you might want to edit my cable page if you happen to like the layout. You can keep the same symbols that I use or edit some that may be different on your device. Then substitute in your own IR commands and protocols.

You could also completely remove an entire page. For example you might want to remove the page for my Sony amplifier surround sound system. You basically just reverse the steps that we used to add a page. Everywhere that we told you to put in a new number or new rows of items you would just delete the old rows. Note that you will have to re-number all of the rows. For example the amp occupies three rows titled \texttt{ROW_AMP_1} through \texttt{ROW_AMP_3}. If you remove those three rows then you should renumber \texttt{ROW_TV_1}, which was \texttt{12}, and make it \texttt{9}. And then continue to renumber every row below that in the entire array.

Special Commands

Of the four parameters in each command, the final one is a flag for special handling. We already showed you how to use the \texttt{SPL_CHANGE_PAGE} special flag. The remaining types of special commands are defined in the main tab of the program at approximately line 114 as follows.

```c
//Types of special commands stored in Current.Special
#define SPL_QUIT 1
#define SPL_REC 2
#define SPL_GOTO_RIGHT 3
#define SPL_GOTO_DOWN 4
#define SPL_GOTO_TVAR 5
#define SPL_CHANGE_PAGE 6
#define SPL_HOLD 7
#define SPL_KB_SHIFT 8
```

We already discussed how the Quit command worked on the opening page. The \texttt{SPL_QUIT} flag initiates that. Also we mentioned that when you select a \texttt{RECORD} function it causes a pop-up screen to ask "Are You Sure?". That is handled by \texttt{SPL_REC}.

The next three \texttt{SPL_GOTO} flags transmits the signal that you've selected but then it automatically moves your cursor to a different location on the screen. We implemented this feature because we noticed that when selecting a function such as Menu the very next thing you want to do is move to the arrow keys to navigate the menu. Similarly the on-screen guide you want to jump to the arrow keys. You can look at the code in the main tab at approximately line 197 and find the function \texttt{do_Special(void)} to see how these work.

The flag \texttt{SPL_HOLD} causes the same IR function to be retransmitted for a couple of seconds. Unlike cable box there are certain functions such as a jump forward 15 minutes that can only be attained by holding the fast for one full second. This special flag implements that. This feature is unique to my particular cable DVR and if you need to implement something similar it might require some extra code in the \texttt{do_Special()} function of the main tab.

Finally the \texttt{SPL_KB_SHIFT} flag is used by the full keyboard screen. It already works and you will not likely need it anywhere else so we will not trouble you with the details.

Optional Features
There are three features that you can turn off or on and recompile and upload the program. We've already mentioned a couple of them but we will review them all here. On the main tab of the program at the very top just below the opening comments you will see the following three lines:

```c
#define MY_DEBUG 0
#define USE_BLE 1
#define USE_TOUCHSCREEN 1
```

Changing the first value from 0 to 1 turns on debugging messages and the serial monitor. You will have to have the serial monitor open or the program will not initialize.

Changing the second line from 1 to 0 turns off all of the Bluetooth functions. If you are not using the ultimate remote for iOS switch control you would not need a Bluetooth processor board such as the Feather M0 BLE. You could substitute a different Feather board such as a Feather M4. The Bluetooth symbol will still appear on the top row of your screen but selecting it will do nothing.

Changing the third value from 1 to 0 will deactivate the touchscreen feature. You may not need it and it may be annoying if you accidentally touch the screen at the wrong time.

**Other Modifications**

We've tried to organize the code in a logical manner and split it up into various tabs to make things as simple as possible. The code is well commented throughout. All of the changes necessary to add or remove pages of commands or to edit commands are all in the file `my_pages.h` so except for the definitions we just mentioned above. You probably will never need to edit any of the other files. If you want to control the device using something other than 3 AT switches, all of the code for those switches is in `my_inputs.h`. You would have to implement new code in that section to read different types of devices such as flex sensors, force sensitive resistors, sip-and-puff devices etc. That portion of the code is well commented. If you need assistance with such modifications feel free to contact us and we will try to help as best we can.
Troubleshooting

Device Freezes When I Select

When you are navigating around the screen if you hit SELECT and the device appears to quit working it probably means that you did not change the default output pin in `IRLibProtocols/IRLibSAMD21.h`. The default output pin on IRLib for Feather M0 boards is pin 9 but that pin is used to control the TFT display. If you fail to change this default, it messes up the display. See the configuration section on how to change the default to pin 12 which is how we wired the device.

Cannot Recognize My IR Device

When you are capturing IR codes from your TV, DVR etc. remote there is a chance that the `dongle_dump` sketch will not recognize it. IRLib supports 11 of the most popular protocols most of which have multiple variants within the protocol that are supported. There is a possibility that your device does not use one of our supported protocols. We will be happy to work with you to try to support your device. Note that IR controlled air conditioners are especially difficult to support and we cannot assist you with that. They use extremely long sequences of pulses that are difficult to decode and store.

Another possibility is that your device uses 57 kHz modulation rather than the typical 38 kHz or variations that range from 36-40 kHz. Even though your TSOP has been built to handle 38 kHz, it should be able to decode anything from 36-40 kHz. But it will not handle 57 kHz. If your device is a cable box or DVR try using the cable codes provided in the sample sketch. My cable box is 57 kHz. It was used by cable boxes built by Scientific-Atlanta, Cisco, Motorola, and Technicolor on cable systems managed by the former company BrightHouse Cable which is now owned by Spectrum Cable. There is only one other 57 kHz protocol that we know of and it is extremely rare. You could purchase a TSOP device capable of reading 57 kHz signals or a TSMP58000 device which is used to detect the frequency of a signal. Details on the use of these alternate devices are available in the IRLib documentation and in the tutorial on the IR transmitter and receiver board.

There are detailed instructions in the IRLib documentation on how to implement your own protocols but that is a very advanced topic. We are committed to working with you to try to support your device. You will need to edit the `dongle_dump` sketch at the second from the bottom line which says:

```cpp
code
myDecoder.dumpResults(false); //Now print results. Use false for less detail
```

Change the `false` to `true` and capture the signal again. The serial monitor will print out a more detailed analysis of your signals. You can email that to me at cy_borg5@cyborg5.com and I will do what I can to assist you.

One other possibility is that your remote does not use infrared signals. Some remotes for example on DirecTV satellite boxes have an option of using either infrared or some sort of RF or Bluetooth wireless system. You can typically reconfigure your satellite box and your remote to use IR signals instead of RF signals. Check out your DirecTV support websites or do a Google search to learn how to convert your remote and your satellite box from RF into IR.

Unfortunately some remote controls do not have an infrared option and we will not be able to control those devices at all. For example if you are using a PlayStation 4 as a media player it does not use infrared.

Other Issues

Make sure that you have downloaded the latest libraries for the and board updates for the Feather M0 BLE and the 2.4" TFT display. Double check your wiring and reread the documentation. If you get really stuck contact me at cy_borg5@cyborg5.com and I will do I can to help.

Final Thoughts
Normally when writing tutorials for the Adafruit Learning System, the projects might be built by hundreds of users and trying to support them individually would be beyond my capabilities. But I understand that this particular device is not going to be used by a lot of people. For those that will use it, it has the potential to have a great impact in their independence and ability to overcome their handicaps. The device has been a lifesaver for me literally. I used the Bluetooth switch control to communicate with doctors and nurses when I could not speak well on a ventilator in the hospital. All of the Fusion 360 design work, all of the programming, and all of the writing of this tutorial including everything the videos was done with the assistance of this Ultimate Remote and dictation software Dragon NaturallySpeaking. That should illustrate how important this device is to people like me. That's why I'm so willing to work with you to customize the device and get the most use of it possible. Even if you do not need any assistance or support, drop me an email at cy_borg5@cyborg5.com and let me know how you use this device. Also drop by the ATMakers.org Facebook page (https://adafruit.it/FMj) for assistance if needed on this or other assistive technology issues. I would love to hear from you.