

IR Hacks

Getting Started Guide Book

Build an infinitely hackable Arduino-compatible shield with projects for remote cloning, gaming, and beyond!





Caution

The completion of the IR Hacks shield requires soldering, a process that if done incorrectly can be dangerous, and components that can overheat. To ensure your safety, please follow these guidelines:

- 1. Maintain a Clean Workspace:** Set up a clutter-free and well-ventilated area for soldering. Clear away any unnecessary items to minimize distractions and potential hazards.
- 2. Handle with Care:** Treat soldering equipment and materials with care and respect. Avoid rough handling or dropping of components to prevent damage or injury.
- 3. Supervision for Minors:** If you are under the legal consent age, it is crucial to have proper supervision from a knowledgeable adult while soldering. This ensures proper guidance and minimizes potential risks.
- 4. Protective Eyewear:** Wear safety goggles or protective eyewear at all times during soldering. These will shield your eyes from any splashes, sparks, or accidental contact with heated components.
- 5. Appropriate Attire:** Dress in appropriate clothing for soldering, preferably made from natural fibers, to minimize the risk of loose clothing catching fire or interfering with your workspace.
- 6. Proper Ventilation:** Ensure adequate ventilation by working in a well-ventilated area or using a fume extractor. Soldering fumes may contain potentially harmful substances, and proper ventilation helps mitigate their impact.
- 7. Burn Prevention:** Be cautious of hot soldering irons and components. Always place the soldering iron in its holder when not in use, and handle it with the utmost care to prevent burns.
- 8. Flux and Lead Handling:** Handle soldering flux and lead-based solder with care. Wash your hands thoroughly after soldering to prevent accidental ingestion or skin contact.
- 9. Mindful Placement:** Be mindful of the placement of soldering equipment and components to prevent tripping hazards or accidental contact with hot surfaces.
- 10. Emergency Preparedness:** Familiarize yourself with the location of fire extinguishers and first aid kits in your workspace. In case of any mishaps, you should be prepared to respond promptly.

By adhering to these safety guidelines, you ensure a secure and enjoyable soldering experience while assembling the IR Hacks shield. Your safety is of utmost importance to us, and we want you to confidently pursue your creative endeavors while taking every precaution necessary.



Contents

Introduction	01
IR Hacks Electronic Materials	02
Resistor Bands	03
IR Hacks Active Component Diagram	04
Solder the Shield	05
Get Connected	09
Code it to Life	10
Download the Arduino IDE	10
Testing your microcontroller	10
Coding Libraries	10
Download the Required library	10
Test your IR Hacks	13
Downloads Section	13
IR Hacks Codes	14
Test Results	16
Example Projects	17
IR Hacks Universal Remote Code	17
Hack Your Remotes	18
Get Prepared	19
Open the Example Code	21
Capture the Signals	22
Edit the Template	27



Introduction



Welcome to the amomii IR Hacks Getting Started Guide! The IR Hacks shield is a versatile and powerful microcontroller shield designed for learners, hobbyists, and makers interested in exploring infrared technology. Equipped with a 4x4 key array, OLED screen, buzzer, IR reader, and IR transmitter, this feature-packed kit simplifies integrating infrared capabilities into your projects. Whether you're cloning remote controls, creating simple video games, or building a functional calculator, the IR Hacks shield provides endless opportunities for learning, experimentation, and creativity. Designed to connect seamlessly with the amomii UNO and other Arduino UNO-style boards, it also serves as an excellent soldering practice kit, making it an ideal educational tool for both beginners and advanced users.

IR Hacks Electronic Materials



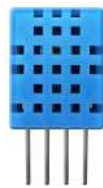
Headers

Female headers (4 Pin) and male headers (40 Pin) are connectors used to make electrical connections between components. On the Neon Tennis shield, they enable easy plug-and-play connectivity between the shield and other components, such as the amomii UNO or other shields.



Tactile Push Button

A momentary switch that allows you to create user input interfaces in electronic projects. On the Neon Tennis shield, these buttons are used for player input, allowing players to interact with the game by simulating the hitting of a tennis ball.



DHT11 Sensor

A digital sensor for measuring temperature and humidity. On the IR Hacks shield, the DHT11 sensor adds environmental sensing capabilities, allowing projects to react to changes in temperature and humidity.



IR Transmitter

A component that emits infrared signals, used to send data to other IR-enabled devices. On the IR Hacks shield, the IR transmitter allows the shield to send infrared signals, enabling it to control devices or communicate with other IR receivers.



Transistor (2N2222A)

A type of NPN bipolar junction transistor used for switching and amplification purposes in electronic circuits. On the IR Hacks shield, the 2N2222A transistor is specifically used to amplify signals for the IR transmitter, ensuring that the infrared signals are strong enough to communicate effectively with other IR-enabled devices.



Buzzer (Passive)

An audio signaling device that produces sound when an electrical signal is applied to it. In the Neon Tennis shield, the buzzer produces sound alerts for various game actions, such as scoring points or signaling the end of a match.



Resistors

Passive components that limit the flow of electrical current in a circuit. On the IR Hacks shield, resistors are used in various parts of the shield to ensure proper current flow and protect components from damage.



XH Connector (4 Pin)

A 4-pin connector used for making secure and reliable electrical connections. On the Neon Tennis shield, the XH connector is used for connecting the shield to the amomii Glow UNO, an optional addition sold separately, enabling additional lighting effects and customization options.



IR Receiver

A component that detects infrared signals, commonly used in remote control systems. On the IR Hacks shield, the IR receiver allows the shield to receive and decode signals from IR remote controls, enabling the cloning of remotes and other IR-based projects.



amomii Blink

An OLED screen module used for displaying information and animations. On the Neon Tennis shield, the amomii Blink module displays animations, scores, and game status, providing players with visual feedback and game information.

Resistor Bands

Resistors come in various shapes and sizes, and have different resistances. The ones included with this kit are all 1/4W 5 band axial resistors and are the same shape and size; however, their resistance ratings vary from 220Ω to 100,000Ω (100kΩ). To avoid potentially damaging the Mini Grand and to insure optimal performance, it is vital to use the correct resistors in the correct places.

To distinguish 5 band axial resistors, you must look at the color of the bands. When oriented the correct way around, the first four bands starting from the left show the resistance value, and the rightmost band shows tolerance (how accurate the resistor is). For our application, precision isn't so important, so we can generally ignore the tolerance band. The resistance bands (first four bands), however, are very important.

Note: You can tell which way to orient the resistor for reading it because the gap between the resistance bands and the tolerance band is slightly bigger.

Now, to read the tolerance value, we look at the color of the bands and the corresponding value of the particular color on the chart. The first three bands are the first three digits in the number, and the fourth band is the multiplier. Basically, for the fourth band, look at the value of the color and add that many zeros to the number.

The resistance bands in the example to the side are brown, black, black and orange, 1, 0, 0, and orange is 3, so add 3 zeros. The resistor to the side has a resistance of 100,000Ω (100kΩ).



See if you can figure out the resistance of the resistor used as the example on the chart.

	Digit	Digit	Digit	Multiplier	Tolerance
Black	0	0	0	x 1	
Brown	1	1	1	x 10	± 1%
Red	2	2	2	x 10 ²	± 2%
Orange	3	3	3	x 10 ³	± 3%
Yellow	4	4	4	x 10 ⁴	± 4%
Green	5	5	5	x 10 ⁵	± 0.5%
Blue	6	6	6	x 10 ⁶	±0.25%
Violet	7	7	7	x 10 ⁷	±0.1%
Grey	8	8	8	x 10 ⁸	±0.05%
White	9	9	9	x 10 ⁹	
Gold				x 10 ⁻¹	± 5%
Silver				x 10 ⁻²	± 10%

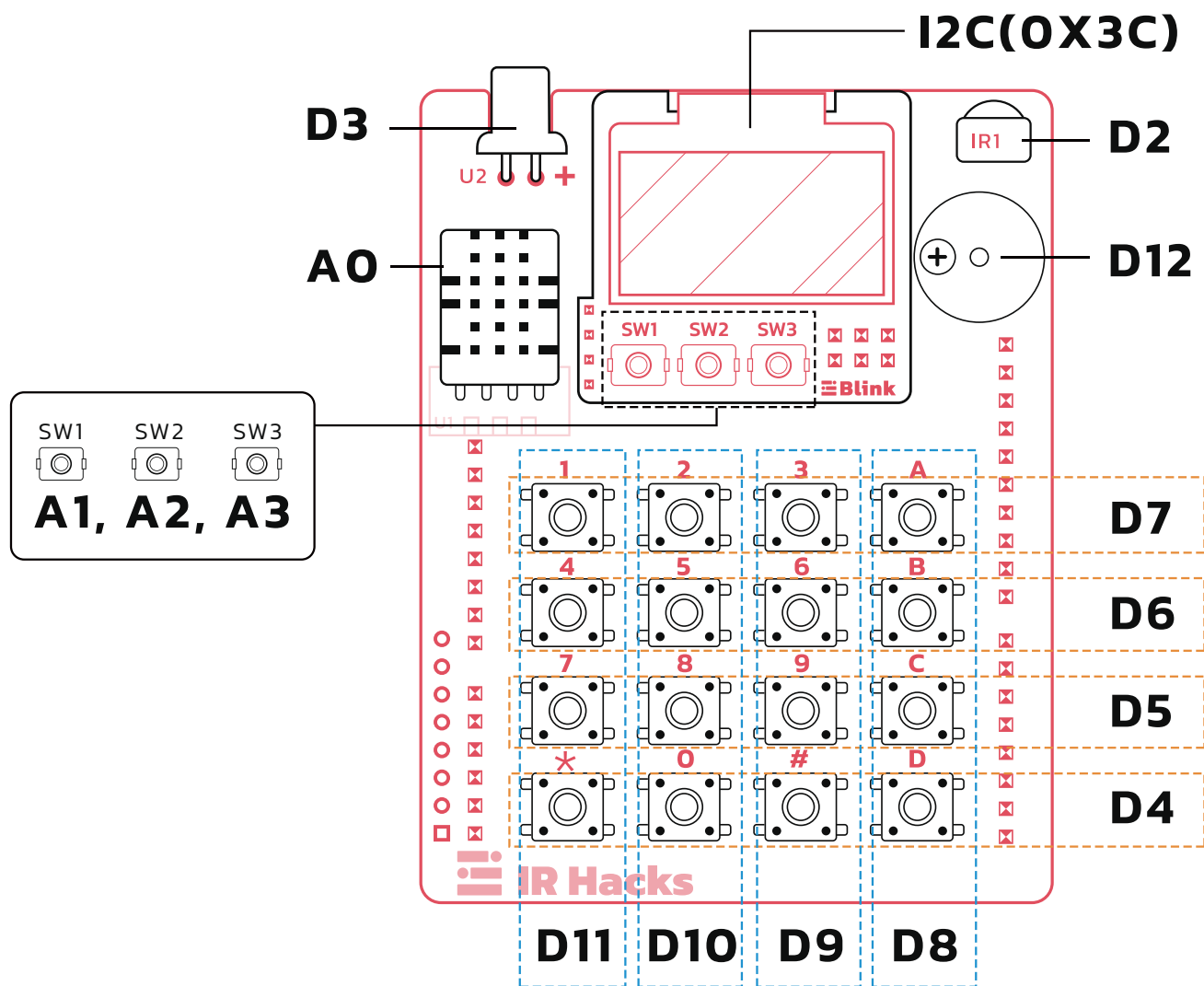
You should have gotten 2000Ω

IR Hacks Active Component Diagram

This diagram highlights the placement of all active components on the IR Hacks shield. It provides vital information, including the corresponding pin connections to your amomii UNO and additional information beneath the diagram. Active components, in this context, refer to components that actively engage with your microcontroller, such as input and output devices. For details on passive components like resistors and capacitors, please consult the Soldering section of this manual.

IR Hacks Pinout

Refer to this diagram to know which components connect to which pins on your UNO board.





Solder the Shield

Prepare your soldering tools



Solder Wire



Soldering Iron



Wire Cutters



Goggles

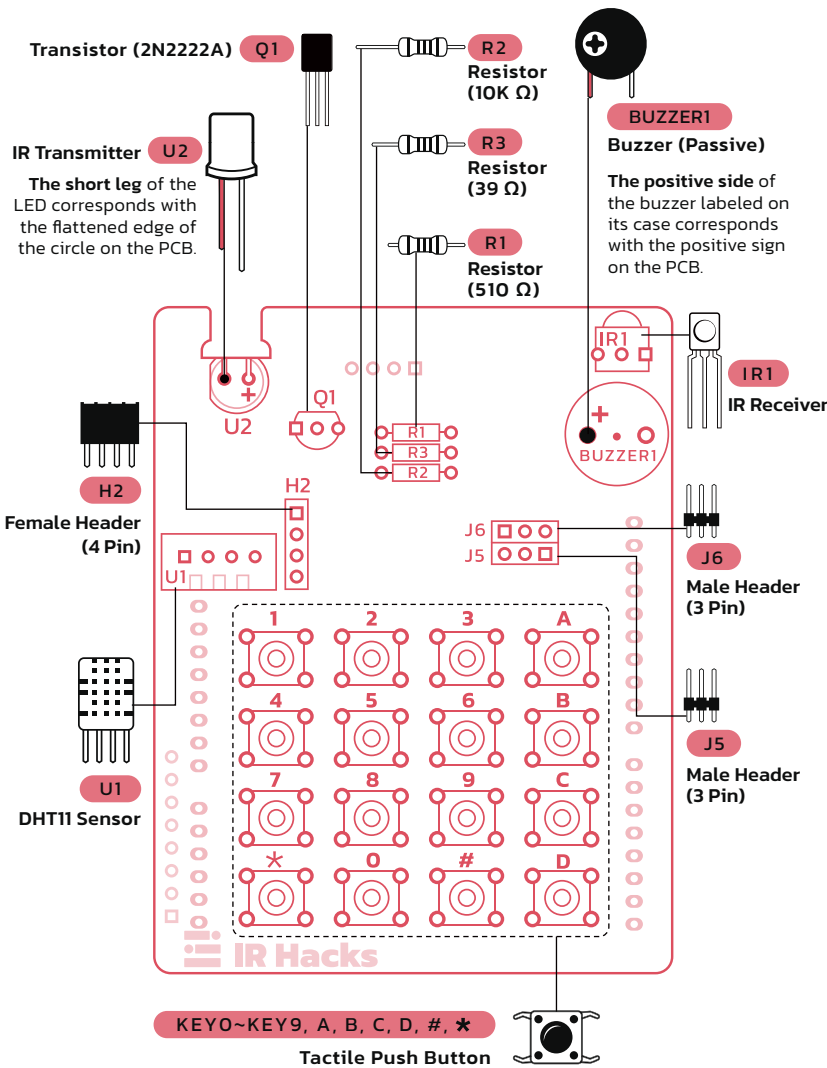


Blu Tack

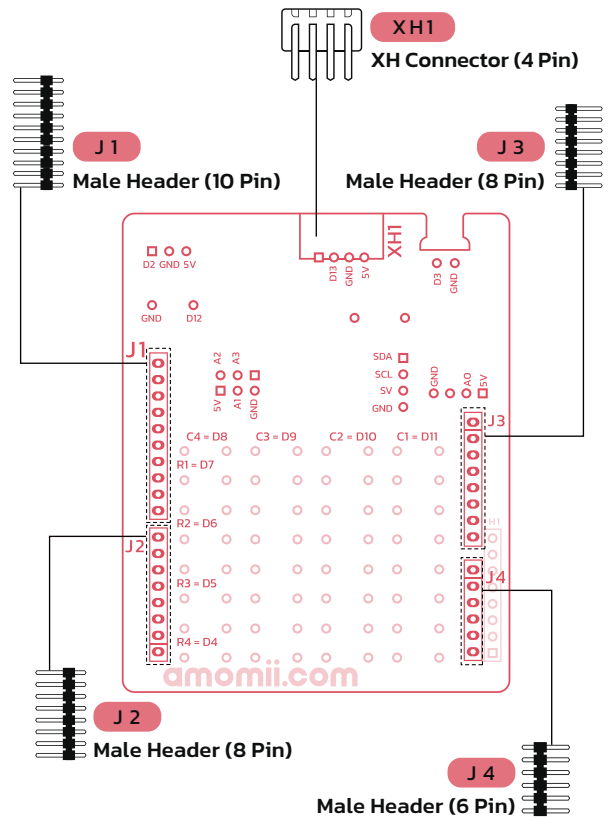


Thick Cardboard




Top Side

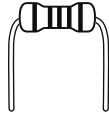
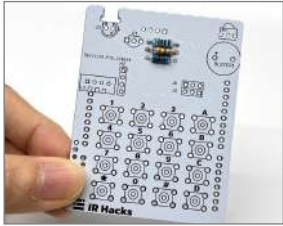


Bottom Side



Step 1

- 1 x Resistor (510 Ω) → RE1 
- 1 x Resistor (39 Ω) → RE3 
- 1 x Resistor (10K Ω) → RE2 



Put all of the resistors in place. Make sure you use the correct resistor by referring to the Materials Diagram on page 1.

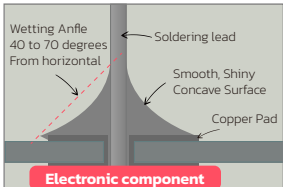
Top Tip: It may be helpful to bend the resistors into U shapes and insert both legs at the same time.



Place a piece of cardboard on top of the resistors to stop them from falling out when you turn over the PCB.



With the PCB lying flat on your table and the resistor legs sticking up towards you, solder each joint, one by one.



When soldering, make sure that the tip of your soldering iron touches and heats up both the leg of the resistor and the solder pad before introducing the solder wire. A properly soldered joint should have a cone shape from the pad to the leg.



Utilize wire cutters to trim the excess legs of the resistors. Please be aware that the cut-off legs may project forcefully, posing potential hazards. Exercise caution by wearing protective goggles to ensure your safety.

Step 2

- 16 x Tactile Push Button → KEY1~ KEY9 
- A, B, C, D, #, *



Install the 16 Tactile Push Buttons on the PCB panel



Up until now, slight alignment issues would only be an aesthetic problem. However, if the push buttons are not flat, it could affect performance, so make sure they are all even and flat before proceeding to solder them in place.



Solder the legs in place.

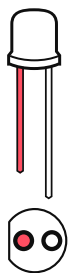


Step 3

- 1x TR Transmitter → U2 



Bend the pins so the IR LED is at a 90 degree angle.



Note: LEDs are polarized, meaning that if they are not connected correctly, they will not work. Make sure the shortest leg (cathode) corresponds to the flat edge of the circular LED symbol.




Insert the IR LED into the shield so that the LED sits nicely in the cutout shape at the top of the board.



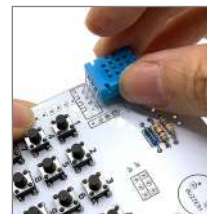
Utilize wire cutters to trim the excess legs off the resistors. Please be aware that the cut-off legs may project forcefully, posing potential hazards. Exercise caution by wearing protective goggles to ensure your safety.

Step 4

- 1 x DHT11 Sensor → U1 



Bend the legs of the DHT11 sensor down 90 degrees.



Insert the legs through the holes with the sensor lying flat against the shield.



Solder the legs in place.



Utilize wire cutters to trim the excess legs off the resistors.

Step 5

1 x Transistor(2N2222A) → Q1



Push the transistor in place. Be Sure that the flat edge of the transistor corresponds to the flat edge of the transistor symbol on the PCB. You will need to slightly spread out the legs of the transistor so they fit in the holes.

Note: You will not be able to make the transistor sit flush with the PCB as you have done with other components, but this will not negatively impact operation. Look at the image for reference.

Place a piece of cardboard on the PCB before you turn it over to prevent the transistor from falling out. With the PCB lying flat on your table and the transistor legs sticking up towards you, solder each joint, one by one.

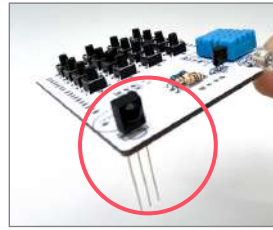


Utilize wire cutters to trim the excess legs off the resistors.



Step 6

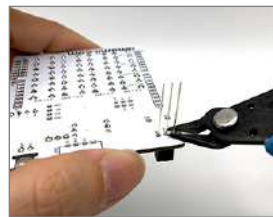
1 x IR Receiver → IR1



Repeat the same steps for the IR Receiver. Note that similar to the transistor, the flat edge of the IR Receiver must also correspond to the flat edge on the PCB.



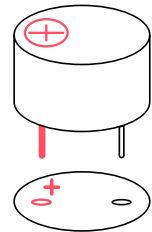
Place a piece of cardboard on the PCB and turn it over to solder the legs in place.



Utilize wire cutters to trim the excess legs off the resistors.

Step 7

1 x Buzzer(Passive) → BUZZER1

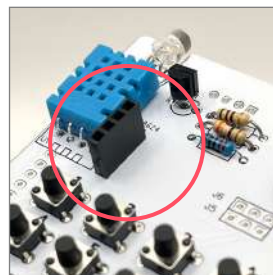


Connect the passive buzzer to the shield. **Be sure that the positive sign on the buzzer corresponds with the positive sign on the buzzer symbol on the PCB.** Turn it over and solder it in place.



Step 8

1 x Female Headers (4 Pin) → H2



Install the 4-pin female header into the position marked H2 on the PCB. Ensure it is also fully seated and flat against the PCB surface.



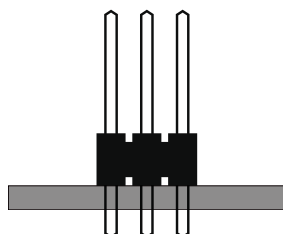
Solder all the pins of the female headers to secure them in place.

Step 9

2 x 3Pin-Male Headers → J5,J6



Snap the pin header strip into 2 sections to create J5 and J6.



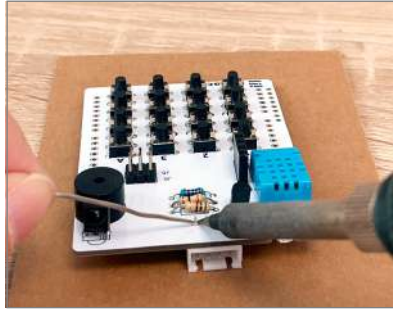
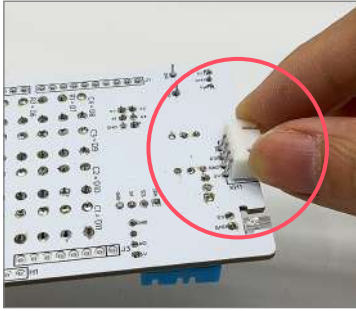
Note: Snap appropriate sized sections from the 40 pin header strip to create J5 and J6, leaving the remaining pins reserved for **Step 11**.

Insert the short side of the pin headers into the positions marked J5 and J6 on the PCB, with the short side protruding through the PCB.

Solder the pins in place to secure them.

Step 10

1 x XH Connector (4 Pin) → XH1



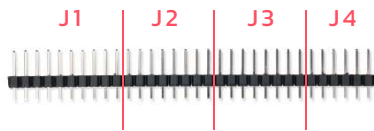
Place the XH Connector in position, turn over the PCB, and solder each leg one by one. Pay careful attention not to bridge any of the legs (make sure none of the legs get soldered together).

Step 11

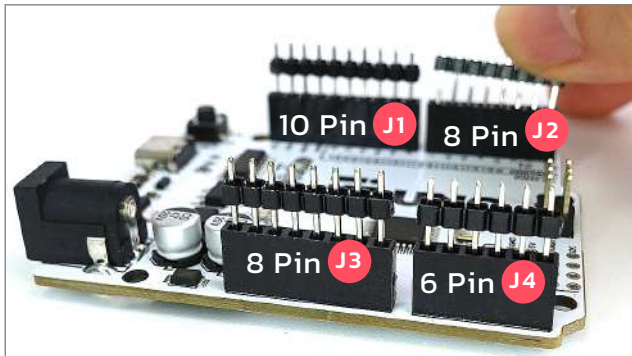
1 X amonii UNO
(or any other Arduino UNO style microcontroller)



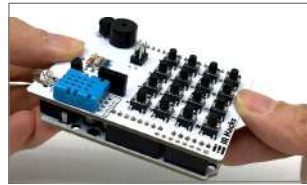
10 Pin-Male Headers → J1
8 Pin-Male Headers → J2
8 Pin-Male Headers → J3
6 Pin-Male Headers → J4



Push the long side of the header pins into the UNO ports as shown in the picture.



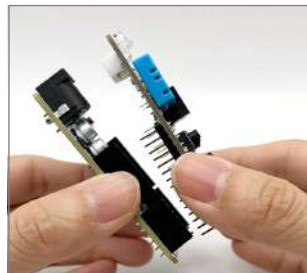
Make sure that the long legs are firmly inserted into the UNO ports with the short pins sticking up.



Next, place the IR Hacks shield on top so that the short header pins protrude through the board and stick out of the top of the PCB. It is often easier to push one side through first and then the other, using the first side as a hinge of sorts.



Solder the pins to the top of the IR Hacks PCB, creating your IR Hacks shield.



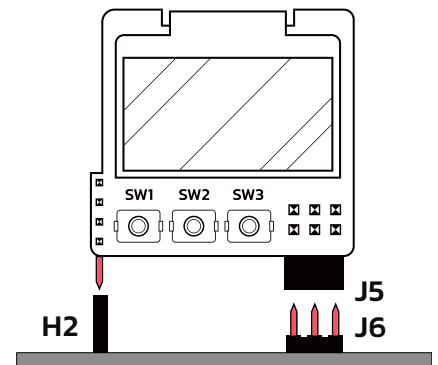
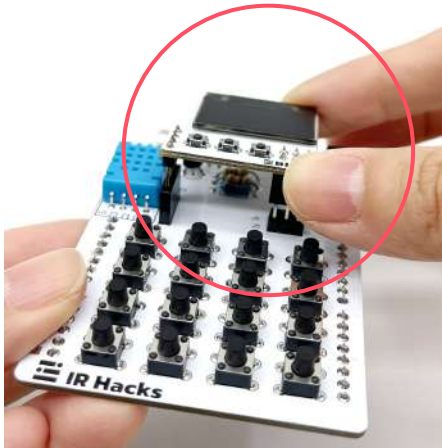
Remove the IR Hacks shield from the UNO. When you pull the two apart, the header pins should stay connected to the IR Hacks shield and come out of the UNO ports.

amonii Blink



Connect the amonii Blink to the shield at pins H2, J5 and J6.

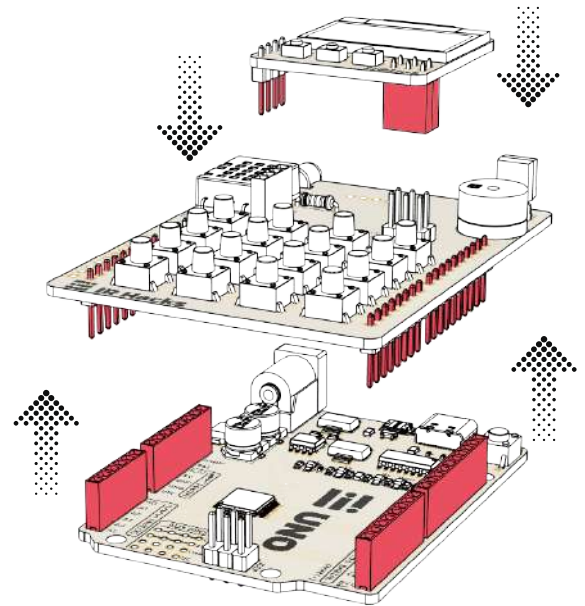
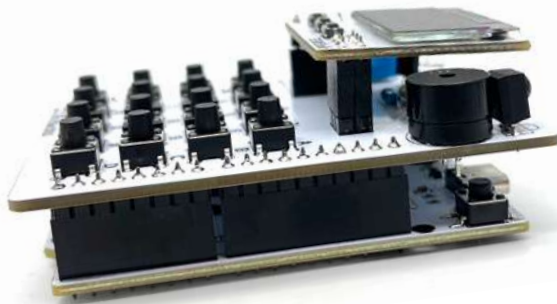
Note: Be sure to handle the module by the sides to avoid damaging the screen.



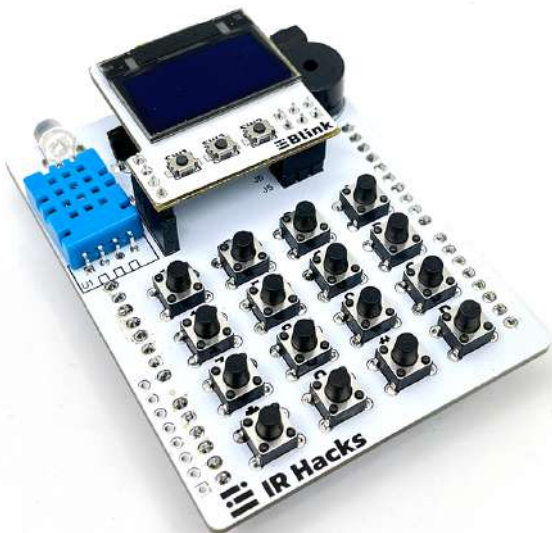


Get Connected

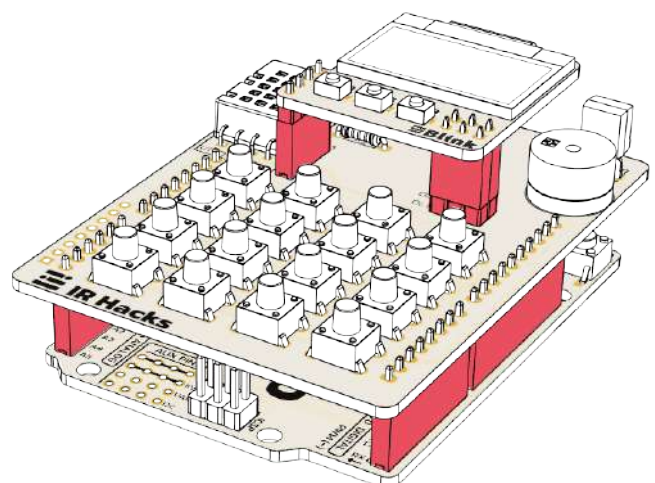
To connect, simply insert the male header pins protruding from the bottom of the IR Hacks shield into the female header pins on the top of the amomii UNO. Make sure that all pins are inserted correctly and be sure to push the IR Hacks shield firmly in place.



If you have not already, connect the amomii Blink to the shield at H2, J5 and J6, ensuring it is firmly in place. The OLED screen on the amomii Blink will be connected via the I2C interface using pins A4 and A5, and the buttons (SW1, SW2, SW3) will be connected to analog pins A1, A2, and A3, respectively.



Make sure all connections are secure to ensure proper functionality of the IR Hacks shield.





Code it to Life

Before you start uploading code to your IR Hacks shield, you will need to download the Arduino IDE and ensure you can use it to upload code to your UNO board. If you already have the Arduino IDE and know your UNO board is working, you can proceed to the Coding Libraries section.

Download the Arduino IDE

The Arduino Integrated Development Environment (IDE) is the software used to write, upload, and debug sketches (programs) on your microcontroller (the amomii UNO, for example). The IDE can be downloaded for free from the Arduino website (www.arduino.cc). The IDE is available for Windows, Mac OS X, and Linux operating systems. Simply go to the "Software" section of the Arduino website, select the appropriate operating system, and follow the instructions for downloading and installing the IDE on your computer.

Testing your microcontroller

If you bought an amomii UNO with this project and have not yet tested it, I suggest that you do this first. Please refer to the [amomii UNO Getting Started guide](#) before proceeding beyond this point and come back when the amomii UNO is tested. If you plan on using any other UNO board, such as the Arduino UNO, please ensure it is working correctly before proceeding beyond this point.

Coding Libraries

Coding libraries are collections of pre-written code that simplify complex tasks. For the IR Hacks shield, we use various libraries to manage the OLED display, IR transmitter, and DHT11 sensor. Specifically, there are two libraries we recommend downloading for the OLED display: [lcdgfx](#) and [Adafruit_SSD1306](#). The [lcdgfx](#) library by Alexey Dynda is optimized for lower resource usage, ensuring smooth operation and compatibility. Meanwhile, the [Adafruit_SSD1306](#) library offers additional features, such as the ability to display bitmaps, making it valuable for certain example sketches. Additionally, the [IRremote](#) library is necessary for controlling the IR components, and the [DHT](#) library is required for reading data from the DHT11 sensor.

Download the Required Libraries

There are various ways to download coding libraries for the Arduino IDE, but the simplest way is to download them directly from within the IDE itself. We will be using Version 2 of the IDE, but the steps for the original are similar.



- 1 Click on the **library manager** shortcut on the left-hand side of the IDE.

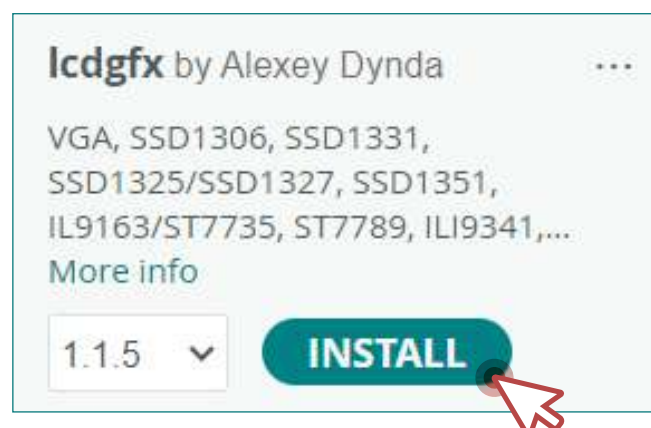
Note:

- a There is no shortcut here on the original Arduino IDE, but the Library Manager can be accessed from **Tools > Manage Libraries**.
- b Some libraries depend on other libraries themselves, so you may be asked whether you want to allow these to be downloaded too. As a general rule of thumb, allow this by selecting **"INSTALL ALL"**.

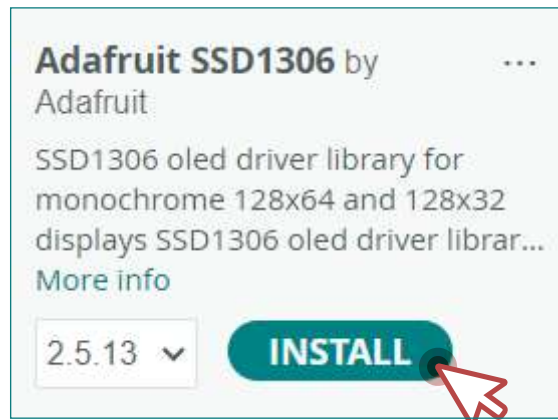


- 2 When the Library Manager pops up, search for the following libraries and install them:

- **lcdgfx**: Search for **lcdgfx** by Alexey Dynda and click **INSTALL**. This library is used for controlling the OLED display (amomii Blink) and is preferred for its lower resource usage.



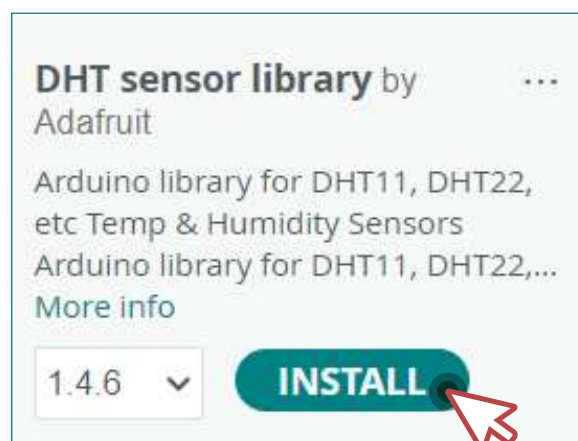
- **Adafruit_SSD1306**: Search for Adafruit_SSD1306 and click **INSTALL**. This library is used in some example sketches for its ease of use and features such as displaying bitmaps. The **Adafruit_GFX** library, required by **Adafruit_SSD1306**, will be installed automatically as a dependency.



- **IRremote**: Search for IRremote and click **INSTALL**. This library is essential for using the IR transmitter and receiver, enabling you to send and receive IR signals.



- **DHT**: Search for DHT by Adafruit and click **INSTALL**. This library is required for reading temperature and humidity values from the DHT11 sensor.



Test your IR Hacks

Now, it's a good idea to run the test code to ensure all components are working correctly and your soldering job was successful. You can download the official amomii IR Hacks codes from several sources: it was emailed to you in your welcome email, and you can also access it from the Downloads section at amomii.com.

Welcome email

If you haven't received your Welcome email, scan this QR code or enter the URL into your browser and tell us where to send the email. Your welcome package includes documentation, guides, code, and more.



- Example Code
- Datasheet
- Getting Started Manual
- Schematic
- 3D Printable STLs

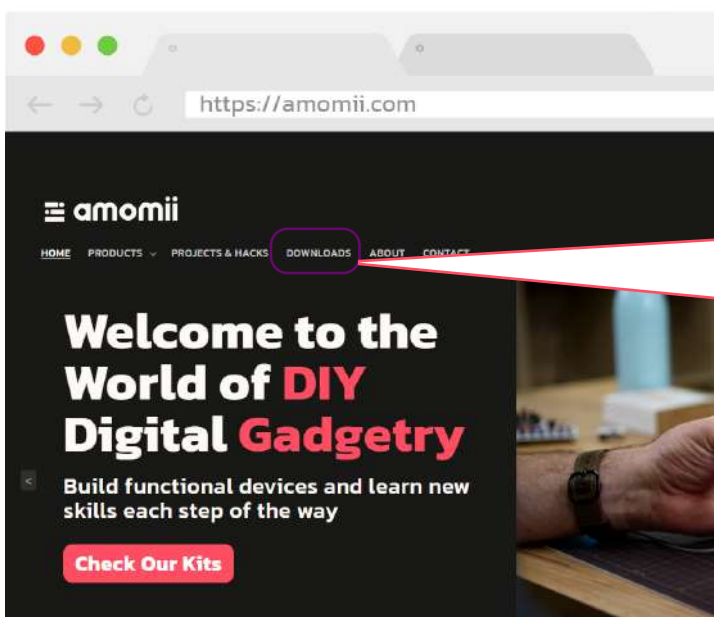
amomii.com/pages/irh_me3se

Downloads Section

To access the code from our website, visit the [Downloads section](#).

NOTE

You will need to create a free account to access this area. Once you are in the Downloads area, locate the **IR Hacks** section and click on the **"Code"** hyperlink. All of the IR Hacks code should start downloading in a ZIP folder.



IR Hacks

- Getting Started
- Datasheet
- Schematic
- **Code**
- 3D Printables

amazon



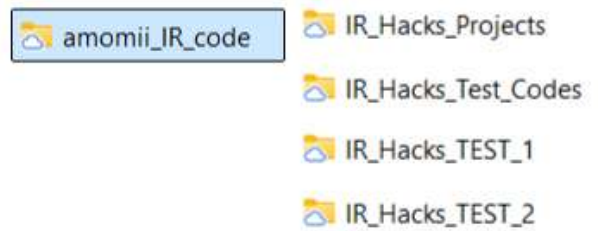
Did you receive your welcome email?

If you bought your IR Hacks kit from Amazon, click here to receive the welcome email containing all of your digital assets. The hardware is just part of the package, click here to claim the rest.

IR Hacks Codes

The downloaded ZIP folder is called 'amomii_IR_code'. Unzip the folder to access the code. It is important not to remove files from the folders as the sketches rely on them and may not work correctly if altered. Inside this folder, you will find two more folders:

IR_Hacks_Projects and **IR_Hacks_Test_Codes**.



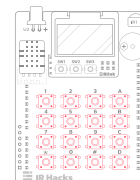
IR_Hacks_TEST_1

Open the IR_Hacks_TEST_1 folder, double click on the INO file of the same name and the Arduino IDE should open with the sketch ready to upload to your device.

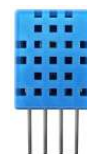
This code tests all the components of the IR Hacks shield except for the IR communication components. It verifies the functionality of the OLED screen (amomii Blink), keypad, buzzer, and DHT11 sensor.

1. Upload the IR_Hacks_TEST_1 code to the UNO with the IR Hacks shield attached.
2. Open the Serial Monitor and set the Baud Rate to **115200**.
3. Follow the on-screen instructions to test each component:

- **Keypad:** You will be prompted to press each key on the 4x4 keypad matrix.



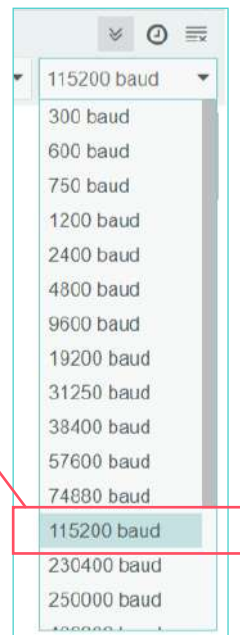
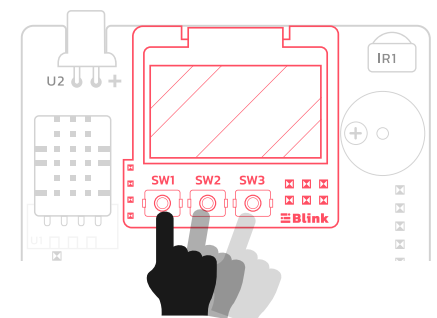
- **DHT11 Sensor:** The test will display temperature and humidity readings, and you will confirm if the readings are accurate.



- **Buzzer:** The test will play a tune to verify the buzzer's functionality.



- **OLED Screen (amomii Blink):** You will be prompted to press buttons **SW1**, **SW2**, and **SW3** to confirm the display and button functionality.

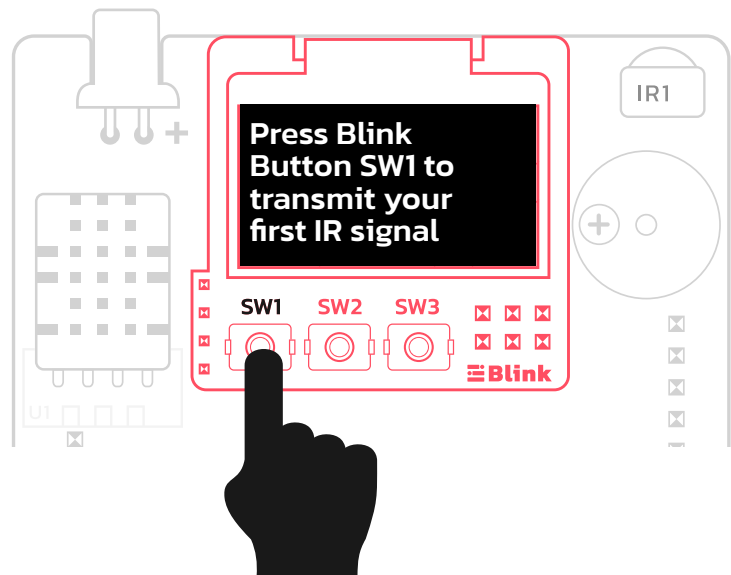


IR_Hacks_TEST_2

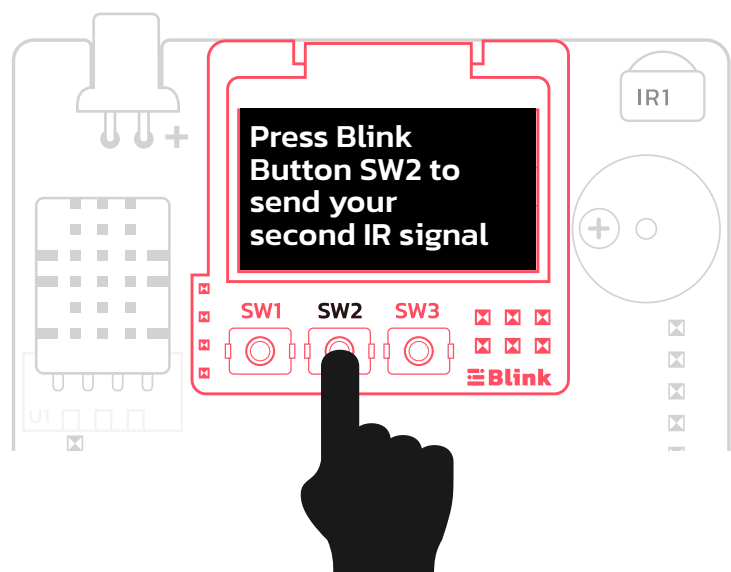
This code tests the IR transmitter and receiver components of the IR Hacks shield. It transmits a signal and checks if the receiver detects it. Using a mirror to reflect the IR signals back at the device may improve reliability.

1. Upload the IR_Hacks_TEST_2 code to the UNO with the IR Hacks shield attached.
2. Open the Serial Monitor and set the Baud Rate to 115200.
3. Follow the on-screen instructions to test the IR components:

- For the first test, **press SW1** to transmit an IR signal and wait for confirmation of reception.



- For the second test, **press SW2** to transmit another IR signal and wait for confirmation of reception.



Test Results

The Serial Monitor will display whether each IR signal was successfully received or not.

If both tests pass, the IR transmitter and receiver are working correctly.

```
-> Time to test the IR Transmitter and Receiver
-> Press Blink Button SW1 to transmit your first IR signal
-> Press detected
-> Waiting for IR signal...
->
-> RECEIVED: PASS
->
-> Press Blink Button SW2 to send your second IR signal
-> Press detected
-> Waiting for IR signal...
->
-> Received. PASS
->
-> If you passed both tests, everything is working and you are ready to enjoy your device!
-> Otherwise, check your soldering and connections and try again!
```

If any test fails, check the soldering and connections and try again.



By running these tests, you can ensure that all components of your IR Hacks shield are functioning correctly before proceeding to your projects.



Example Projects

There are several example projects in the code file you downloaded previously, but in this guide, we will focus on the IR Hacks Universal Remote. You can find this in the amomii_IR_Code folder in the IR_Hacks_Projects folder. Find the IR_Hacks_UNI_Remote folder, open it and double click on the INO file of the same name and the IDE should open up. Once it is open, follow the steps below to edit the template to get it working for your remotes.

IR Hacks Universal Remote Code

The IR Hacks Universal Remote Code is a versatile and customizable sketch designed for use with the amomii IR Hacks shield. This code enables you to control multiple devices with different IR protocols, all from a single device. By leveraging the capabilities of the IR Hacks shield, you can create a universal remote that can switch between up to four different remotes, each capable of sending IR signals to various devices.

Key Features

- **Multi-Remote Support:** Control up to four different devices, such as TVs, air conditioners, or RGB light strips, by switching between remotes using the keypad.
- **OLED Display:** The integrated OLED display shows the name of the currently selected remote, making it easy to know which device you are controlling.
- **Temperature and Humidity Display:** The DHT11 sensor allows you to view real-time temperature and humidity readings on the OLED display by pressing a button.
- **User Customizable:** The code is a template that users can customize by adding their specific IR signals, found using the IRremote library's example code.

Libraries Used

- **IRremote:** For sending and receiving IR signals.
- **lcdgfx:** For controlling the OLED display.
- **DHT:** For reading data from the DHT11 temperature and humidity sensor.

This guide will walk you through the process of setting up your hardware, uploading the code, and customizing it to create your own universal remote.

By the end of this guide, you will have a fully functional universal remote, capable of controlling multiple devices with ease.

NOTE

This project supports most common IR communication protocols, such as Sony, Panasonic, NEC, etc., and should work with the majority of remotes. However, there may be some remotes that do not function as expected. This could be due to the remote using a new or uncommon IR communication protocol not supported by the IRremote library, or employing other wireless communication methods such as Bluetooth or Wi-Fi. Have fun experimenting with different remotes, and get your hack on!

This project can be broken up into two sections:

1. Hack Your Remotes – Deciding which remotes and specific buttons you want to clone, which buttons on the IR Hacks shield you want to assign them to, and to run the code to receive and decode IR signals.
2. Edit the Template – Take the information you noted down from the Hack Your Remotes section, and use it to modify the IR_Hacks_UNI_Remote code template.

Hack Your Remotes

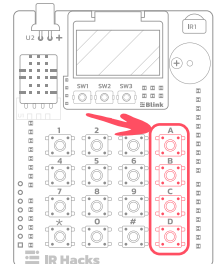
The first step is to choose the remote controls and the buttons you want to hack (clone).

In my example, I will clone three remotes to keep the instructions concise, but you can clone up to 4 different remotes by just repeating the step for each remote.

I will use the remote for two of my TVs and the remote for the RGB LED lighting behind one of them

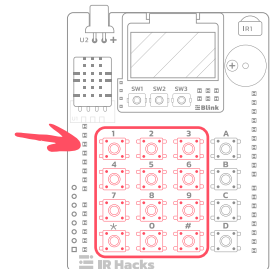
IR Hacks Buttons

On your finished Universal Remote, the rightmost column on the IR Hacks keypad is used for selecting your different cloned remotes, Remote **A**, **B**, **C** or **D**.

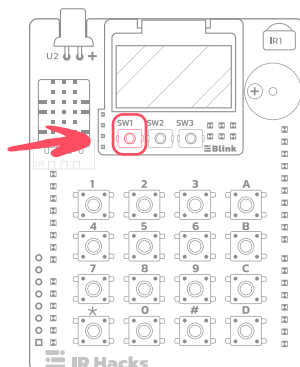


The remaining **12 buttons** on the keypad can be assigned commands from your various remotes

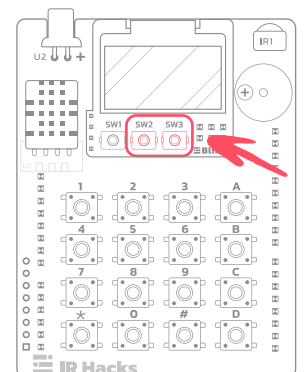
This means that up to **12 buttons** from each remote can be cloned; it is, in effect, a 48 button remote control.



Additionally, **SW1**, the leftmost button on the amomii Blink OLED module can be pressed to display the current temperature and humidity.



Leaving **SW2** and **SW3** functionless and available for any cool additions to the project you may want to make.



Get Prepared

Next, we need to decide which remotes to use, the buttons we want to clone, and the IR Hacks buttons to assign commands to. It's important to make a clear record of this information.

You can organize your information using various methods, such as Word, Excel, Google Docs, or even a pen and paper. For this guide, I will use separate tables for each remote on Google Docs.

Here is a template of the table for Remote A:

Remote A:			
Remote Button	IR Hacks Button	Code	Number of Repeats

I will then fill in the information I can for each remote, leaving the "Code" and the "Number of Repeats" columns empty for now.

I will use the remote for the TV in my Sitting Room as Remote A, cloning some of the buttons I most commonly use.:

Remote A: Sitting Room TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1		
Up	2		
Down	8		
Left	4		
Right	6		
Select	5		
Vol Up	7		
Vol Down	*		

For Remote B, I will use the remote for the TV in my lounge. I use the same common buttons as Remote A, so I will use the same table as the previous one, only changing the title for now. However, As the TVs are different brands, the code will be different when I come to filling that in.

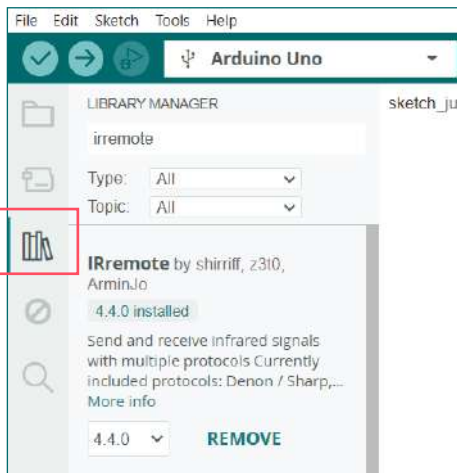
Remote B: Lounge TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1		
Up	2		
Down	8		
Left	4		
Right	6		
Select	5		
Vol Up	7		
Vol Down	*		

Finally, the lounge TV (Remote B) has RGB lighting behind, so I will clone that as Remote C.

Remote C: Lounge TV Color			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1		
Red	2		
Green	3		
Blue	4		
Purple	5		
Cyan	6		
Yellow	8		
Brightness UP	7		
Brightness DOWN	*		

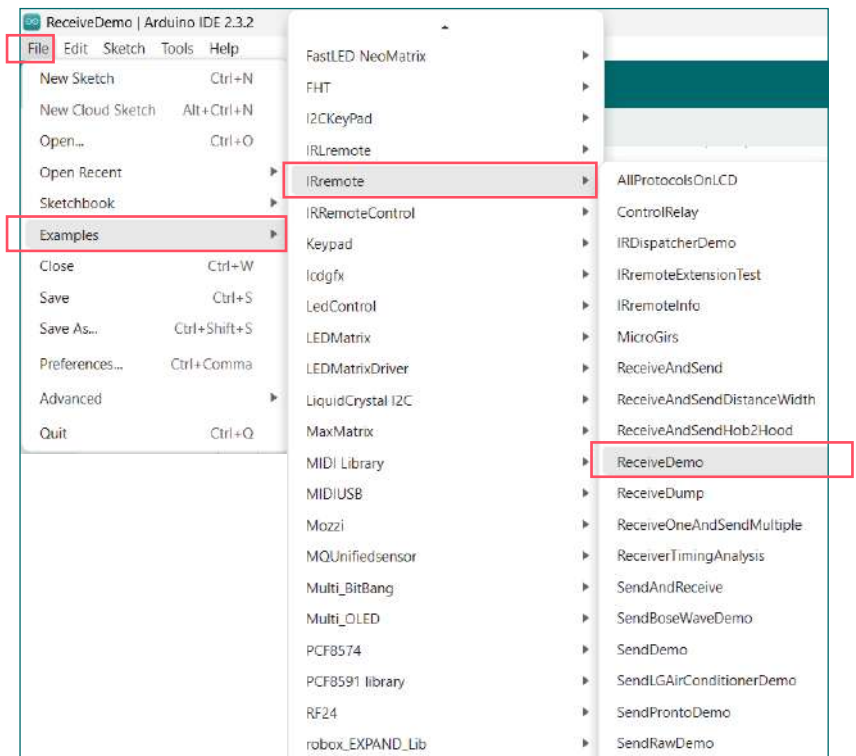
These three remotes work as good examples because we have two remotes that we want to have the same functionality, but are made by different manufactures and have different IR commands and protocols, and a remote for controlling a completely different type of device in the RGB LED strip.

Open the Example Code



Now it's time to do some hacking.

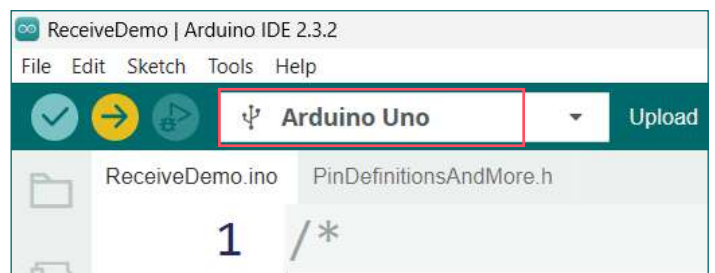
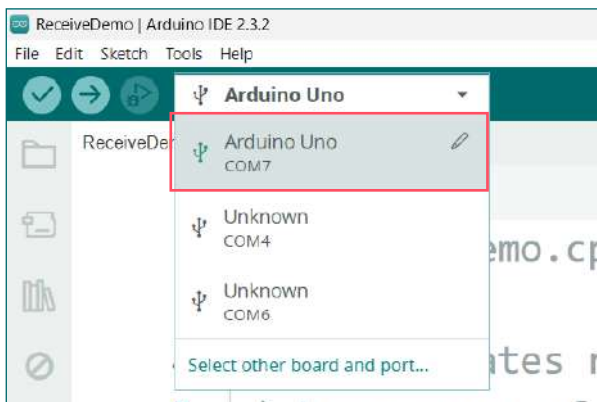
First, ensure that you have downloaded the IRremote library.



Now, go to

File > Examples > IRremote > ReceiveDemo

With your IR Hacks shield connected to your UNO board and the UNO connected to your computer, select the board and COM Port, then upload the code.





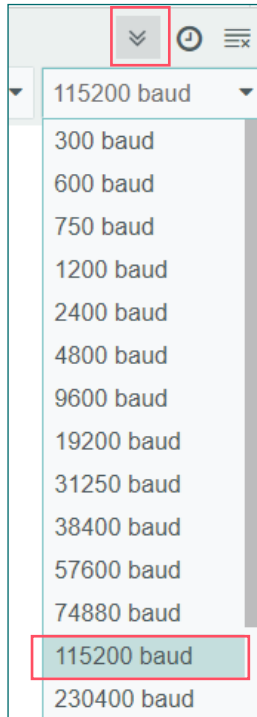
Next, open the Serial Monitor by clicking the magnifying glass found at the top right of the IDE.

The Serial Monitor should open at the bottom of the IDE.

Once it's open, you may see unreadable text printed across the screen like this.



To prevent this from happening, you must change the Baud Rate to 115200 by selecting it from the dropdown menu on the Serial Monitor.



Once the Baud Rate has been corrected, you should be good to go.

Capture the Signals

Point Remote A at the top of the IR Hacks module and press one of the buttons you want to clone.

Looking back at the table for Remote A, I want to clone the power button, so I will first press that. As you can see, when I press it, some information appears in the serial monitor.



```
Protocol=Panasonic Address=0x8 Command=0x3D Raw-Data=0xBD3D0080 48 bits LSB first
Send with: IrSender.sendPanasonic(0x8, 0x3D, <numberOfRepeats>);
```

Try to press the button quickly, without holding it down. In doing so, a second line should appear giving you the code to send when it comes to making our Universal Remote. This line starts with the words **“Send with:”**.

```
Send with: IrSender.sendRC5(0x1, 0xC, <numberOfRepeats>);
```

This is the line we are interested in. Copy the function after the words **“Send with:”** and paste it into your table.

Remote A: Sitting Room TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1	IrSender.sendPanasonic(0x8, 0x3D, <numberOfRepeats>);	

The last column of the table, **“Number of Repeats”**, refers to how many times a signal should be sent per button press. Typically, IR signals do not need to be repeated, but there are exceptions. Protocols such as Sony and JVC often need to be repeated 3 times. This is written into the protocol as a way to improve accuracy. Therefore, if the signal is not repeated, it will not be registered.

Generally speaking, a value of 0 should be put into the **‘Number of Repeats’** column and only changed if the specific protocol used by your remote requires.

Finally, take the value from the “Number of Repeats” column and replace the **<numberOfRepeats>** placeholder in the IrSender function. For most remotes, this value should be 0. For example, if your code is **IrSender.sendRC5(0x1, 0xC, <numberOfRepeats>);** and your **“Number of Repeats”** is **0**, you will update it to **IrSender.sendRC5(0x1, 0xC, 0);**. This completes the code for that specific button.

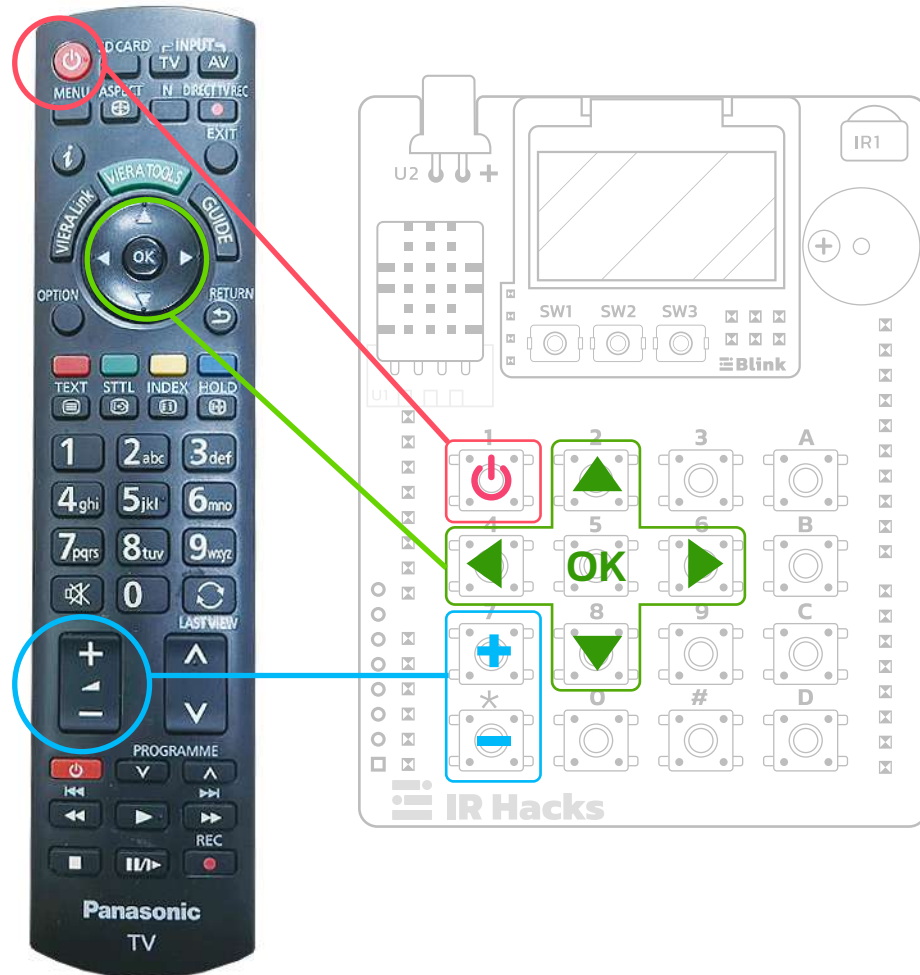
A finished row in a table should look something like this:

Remote A: Sitting Room TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1	IrSender.sendRC5(0x1, 0xC, 0);	0

Next, you must **repeat this for each button on each remote.**

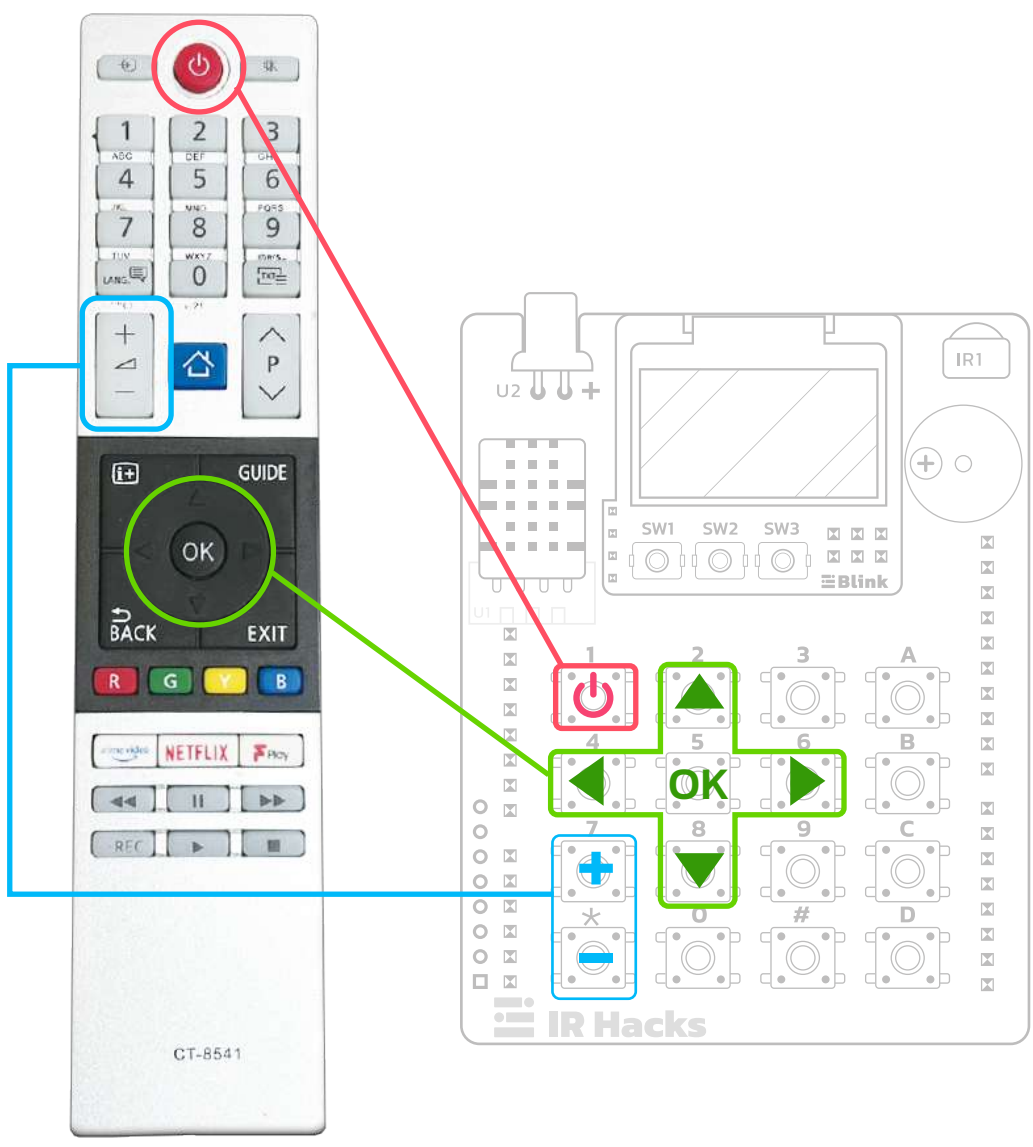
Below, you can see my completed tables next to images labeling the buttons. This diagram has just been added for clarity, you do not need to make a labeled diagram. You should, however, complete a table for each remote before proceeding to the "Edit the Template" section.

Remote A Sitting Room TV



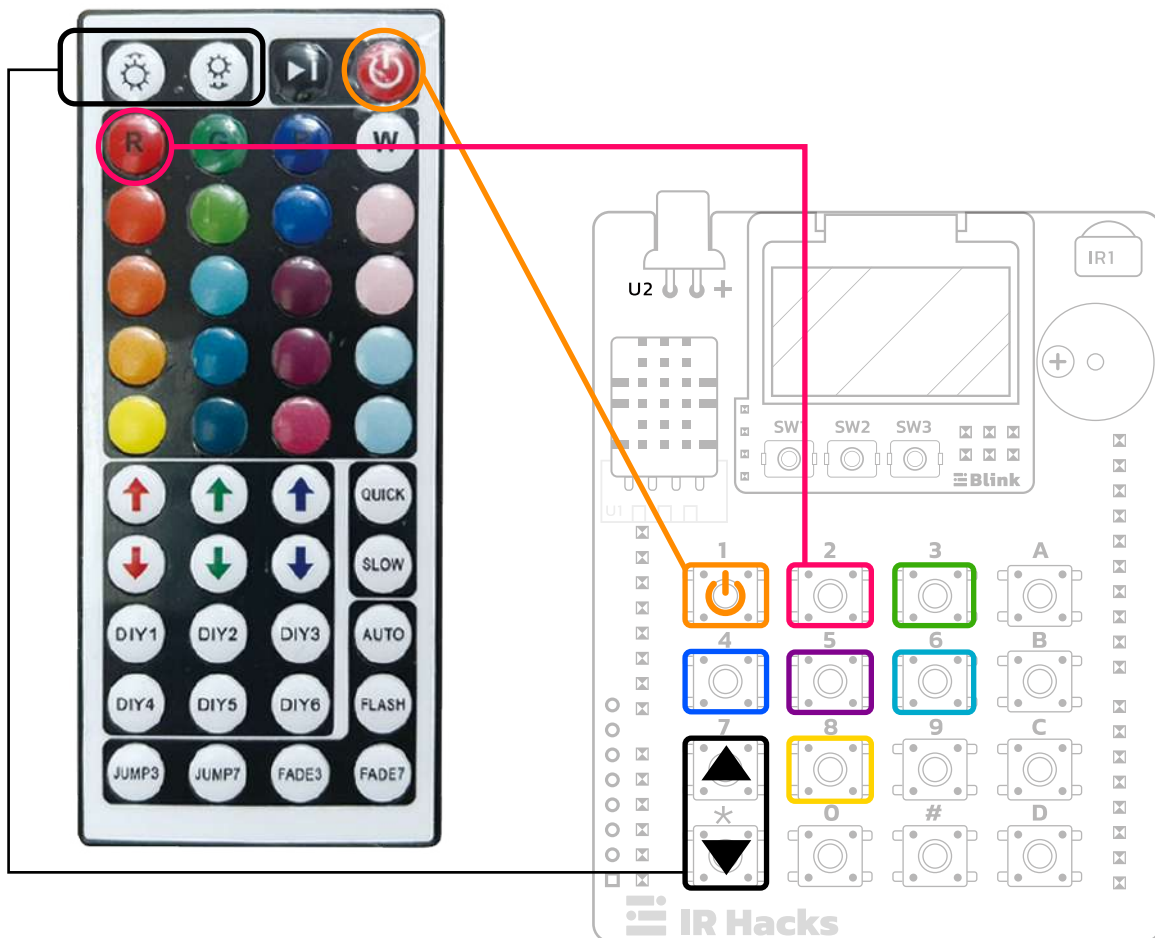
Remote A: Sitting Room TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1	<code>IrSender.sendPanasonic(0x8, 0x3D, 0);</code>	0
Up	2	<code>IrSender.sendPanasonic(0x8, 0x4A, 0);</code>	0
Down	8	<code>IrSender.sendPanasonic(0x8, 0x4B, 0);</code>	0
Left	4	<code>IrSender.sendPanasonic(0x8, 0x4E, 0);</code>	0
Right	6	<code>IrSender.sendPanasonic(0x8, 0x4F, 0);</code>	0
Select	5	<code>IrSender.sendPanasonic(0x8, 0x49, 0);</code>	0
Vol Up	7	<code>IrSender.sendPanasonic(0x8, 0x20, 0);</code>	0
Vol Down	*	<code>IrSender.sendPanasonic(0x8, 0x21, 0);</code>	0

Remote B Lounge TV



Remote B: Lounge TV			
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1	<code>IrSender.sendNEC(0x40, 0x12, 0);</code>	0
Up	2	<code>IrSender.sendNEC(0x40, 0x19, 0);</code>	0
Down	8	<code>IrSender.sendNEC(0x40, 0x1D, 0);</code>	0
Left	4	<code>IrSender.sendNEC(0x40, 0x42, 0);</code>	0
Right	6	<code>IrSender.sendNEC(0x40, 0x40, 0);</code>	0
Select	5	<code>IrSender.sendNEC(0x40, 0x21, 0);</code>	0
Vol Up	7	<code>IrSender.sendNEC(0x40, 0x1A, 0);</code>	0
Vol Down	*	<code>IrSender.sendNEC(0x40, 0x1E, 0);</code>	0

Remote C Lounge TV Color



Remote C: Lounge TV Color

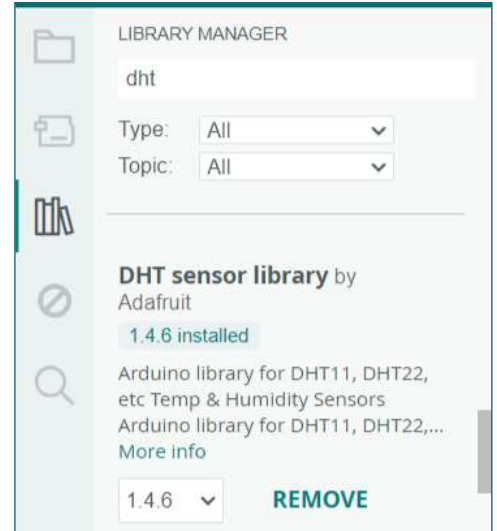
Remote Button	IR Hacks Button	Code	Number of Repeats
Power	1	<code>IrSender.sendNEC(0x0, 0x40, 0);</code>	0
Red	2	<code>IrSender.sendNEC(0x0, 0x58, 0);</code>	0
Green	3	<code>IrSender.sendNEC(0x0, 0x59, 0);</code>	0
Blue	4	<code>IrSender.sendNEC(0x0, 0x45, 0);</code>	0
Purple	5	<code>IrSender.sendNEC(0x0, 0x4D, 0);</code>	0
Cyan	6	<code>IrSender.sendNEC(0x0, 0x51, 0);</code>	0
Yellow	8	<code>IrSender.sendNEC(0x0, 0x18, 0);</code>	0
Brightness UP	7	<code>IrSender.sendNEC(0x0, 0x5C, 0);</code>	0
Brightness DOWN	*	<code>IrSender.sendNEC(0x0, 0x5D, 0);</code>	0

Edit the Template

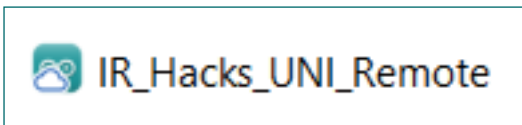
With your tables complete, you are now ready to customize the IR_Hacks_UNI_Remote template to work with your specific remotes. Follow these steps to edit the template code:

● Download the final library

You last library you need to download (providing that you downloaded the ones previously used in the guide) is the DHT library by Adafruit. To download this library, you must search for 'DHT' in the Library Manager and scroll down until you find the one by Adafruit. Install that and you are all set.



● Open the Template Code:



Open the IR_Hacks_UNI_Remote.ino file in your Arduino IDE. This code can be found in the IR_Hacks_Projects folder that you downloaded previously. This template will serve as the base for your universal remote.

● Edit the Remote Names

By default, the remotes are named Remote A, Remote B, Remote C and Remote D, but you can personalize them by changing the words in the remoteNames[] array.

Default Code

```
// Define remote names
const char* remoteNames[4] = { "Remote A", "Remote B", "Remote C", "Remote D" };
```

Example Edited Code

```
// Define remote names
const char* remoteNames[4] = { "Sitting Room TV", "Lounge TV", "Lounge TV Color", "Remote D" };
```

● Locate the Remote Functions:

Scroll down to the sections in the code labeled `remote_1`, `remote_2`, `remote_3`, and `remote_4`. These functions correspond to the four remotes you can control with your shield.

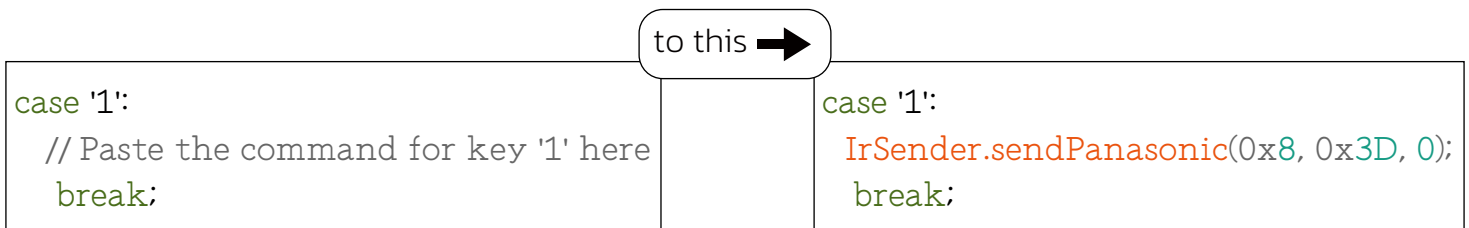
<pre>void remote_1(char key) { ... }</pre>	<pre>void remote_2(char key) { ... }</pre>	<pre>void remote_3(char key) { ... }</pre>	<pre>void remote_4(char key) { ... }</pre>
--	--	--	--

● Assign Commands to Buttons:

Using the data from your tables, replace the placeholder comments within each case statement in the `remote_X` functions with the appropriate IR send commands. Here is how you should do it:

1. Find the case statement for the button you want to program.
2. Replace the placeholder comment with the `IrSender` function you obtained from the `ReceiveDemo`.

For example, if your table for Remote A lists the power button code as "`IrSender.sendPanasonic(0x8, 0x3D, 0);`", go the `remote_1` section, change the case '1' statement from this



NOTE

It is a good idea to leave comments to remind yourself of the function of the button. To leave a comment, you must first enter a double forward slash followed by your comment. To sum up, the code for the first button should look like this.

```
case '1':  
    IrSender.sendPanasonic (0x8, 0x3D, 0); //POWER  
    break;
```

● Repeat for All Remotes:

Repeat the process for `remote_2`, `remote_3`, and `remote_4` functions, ensuring each button's command is correctly entered based on your tables. For any buttons or remotes that you aren't using, just leave them untouched. You only need to edit what you are using.

Here are the complete sections for my example remotes, and remote 4 which is untouched.

Remote A: Sitting Room TV

```
void remote_1(char key) {
    switch (key) {
        case '0':
            //
            break;
        case '1':
            IrSender.sendPanasonic(0x8, 0x3D, 0); //POWER
            break;
        case '2':
            IrSender.sendPanasonic(0x8, 0x4A, 0); //UP
            break;
        case '3':
            // Paste the command for key '3' here
            break;
        case '4':
            IrSender.sendPanasonic(0x8, 0x4E, 0); //LEFT
            break;
        case '5':
            IrSender.sendPanasonic(0x8, 0x49, 0); //SELECT
            break;
        case '6':
            IrSender.sendPanasonic(0x8, 0x4F, 0); //RIGHT
            break;
        case '7':
            IrSender.sendPanasonic(0x8, 0x20, 0); //VOL UP
            break;
        case '8':
            IrSender.sendPanasonic(0x8, 0x4B, 0); //DOWN
            break;
        case '9':
            // Paste the command for key '9' here
            break;
        case '*':
            IrSender.sendPanasonic(0x8, 0x21, 0); //VOL DOWN
            break;
        case '#':
            // Paste the command for key '#' here
            break;
    }
}
```


Remote B: Lounge TV

```
void remote_2 (char key) {
    switch (key) {
        case '0':
            break ;
        case '1':
            IrSender .sendNEC (0x40 , 0x12 , 0); //POWER
            break ;
        case '2':
            IrSender .sendNEC (0x40 , 0x19 , 0); //UP
            break ;
        case '3':
            // Paste the command for key '3' here
            break ;
        case '4':
            IrSender .sendNEC (0x40 , 0x42 , 0); //LEFT
            break ;
        case '5':
            IrSender .sendNEC (0x40 , 0x21 , 0); //SELECT
            break ;
        case '6':
            IrSender .sendNEC (0x40 , 0x40 , 0); //RIGHT
            break ;
        case '7':
            IrSender .sendNEC (0x40 , 0x1A , 0); //VOL UP
            break ;
        case '8':
            IrSender .sendNEC (0x40 , 0x1D , 0); //DOWN
            break ;
        case '9':
            // Paste the command for key '9' here
            break ;
        case '*':
            IrSender .sendNEC (0x40 , 0x1E , 0); //VOL DOWN
            break ;
        case '#':
            // Paste the command for key '#' here
            break ;
    }
}
```

Remote C: Lounge TV Color

```
void remote_3 (char key) {
    switch (key) {
        case '0':
            // Paste the command for key '0' here
            break ;
        case '1':
            IrSender .sendNEC (0x0, 0x40, 0); //POWER
            break ;
        case '2':
            IrSender .sendNEC (0x0, 0x58, 0); //RED
            break ;
        case '3':
            IrSender .sendNEC (0x0, 0x59, 0); //GREEN
            break ;
        case '4':
            IrSender .sendNEC (0x0, 0x45, 0); //BLUE
            break ;
        case '5':
            IrSender .sendNEC (0x0, 0x4D, 0); //PURPLE
            break ;
        case '6':
            IrSender .sendNEC (0x0, 0x51, 0); //CYAN
            break ;
        case '7':
            IrSender .sendNEC (0x0, 0x5C, 0); //BRIGHT UP
            break ;
        case '8':
            IrSender .sendNEC (0x0, 0x18, 0); //YELLOW
            break ;
        case '9':
            // Paste the command for key '9' here
            break ;
        case '*':
            IrSender .sendNEC (0x0, 0x5D, 0); //BRIGHT DOWN
            break ;
        case '#':
            // Paste the command for key '#' here
            break ;
    }
}
```

Remote D: (Unused)

```
void remote_4(char key) {
    switch (key) {
        case '0':
            // Paste the command for key '0' here
            break;
        case '1':
            // Paste the command for key '1' here
            break;
        case '2':
            // Paste the command for key '2' here
            break;
        case '3':
            // Paste the command for key '3' here
            break;
        case '4':
            // Paste the command for key '4' here
            break;
        case '5':
            // Paste the command for key '5' here
            break;
        case '6':
            // Paste the command for key '6' here
            break;
        case '7':
            // Paste the command for key '7' here
            break;
        case '8':
            // Paste the command for key '8' here
            break;
        case '9':
            // Paste the command for key '9' here
            break;
        case '*':
            // Paste the command for key '*' here
            break;
        case '#':
            // Paste the command for key '#' here
            break;
    }
}
```

● Upload the Code:

Once you have entered all the commands, save your changes and upload the code to your IR Hacks device.

● Test Your Remote:

Test each button to ensure it sends the correct command. If any button does not work as expected, double-check the command and the number of repeats.

The images below show me using my Universal Remote to change the volume on my lounge TV, select Netflix on my Sitting Room TV, change the color of the LED strip, and check the temp/humidity by pressing button SW1.

● Changing the Volume



● Selecting Netflix



● Changing the Color



● Checking Temp/Humidity





Congratulations on completing the IR Hacks Universal Remote project! By following this guide, you have successfully customized your own universal remote, capable of controlling multiple devices with different IR protocols. Not only have you gained practical experience with IR communication and Arduino programming, but you've also unlocked a new level of convenience in managing your electronic devices.

Remember, this project is just the beginning. The skills and knowledge you've acquired can be applied to numerous other projects and customizations. Whether you're adding more devices, experimenting with new functionalities, or integrating additional sensors, the possibilities are endless.

We encourage you to share your creations and experiences with the community. If you encounter any challenges or have innovative ideas, don't hesitate to reach out. The IR Hacks community is always here to support and inspire.

You can find links to our social media platforms here:

 amomii.com  Social Platforms

Keep exploring, keep hacking, and most importantly, have fun!

