# Lab 13 – SBB IR Transmitter & Receiver

## Lab Overview

This lab builds upon the foundational knowledge of infrared (IR) communication introduced in the previous lab. Students will explore both transmission and reception of IR signals using the Raspberry Pi Pico, an IR transmitter, and an IR receiver. The lab will focus on setting up a complete IR communication system, encoding and sending IR signals using an IR LED transmitter, and receiving and decoding these signals with an IR receiver. This comprehensive approach aims to deepen students' understanding of IR communication mechanisms, protocols, and practical applications in real-world scenarios.

## Learning Outcomes

Upon completion of the lab, students will be able to:

* Design and Build an IR Transmission System: Construct a circuit using the Raspberry Pi Pico that can transmit IR signals.
* Encode IR Signals: Learn to encode data into IR signals using NEC protocols.
* Transmit and Receive IR Signals: Successfully transmit IR signals from the IR transmitter setup and receive them using the IR receiver setup.
* Apply IR Signals: Implement simple control logic to perform actions based on the decoded IR commands, simulating real-world IR application scenarios.

Table of Contents

[Lab 13 – SBB IR Receiver & programming 1](#_Toc160752285)

[1.1. Lab Overview 1](#_Toc160752286)

[1.2. Learning Outcomes 1](#_Toc160752287)

[1.3. Equipment’s & Materials 3](#_Toc160752288)

[Raspberry Pi Pico 3](#_Toc160752289)

[IR Receiver Module (TSOP382 from Adafruit) 3](#_Toc160752290)

[Breadboard 3](#_Toc160752291)

[Resistors 3](#_Toc160752292)

[LED 3](#_Toc160752293)

[Jumper Wires 3](#_Toc160752294)

[USB Cable: 3](#_Toc160752295)

[Visual Studio Code 3](#_Toc160752296)

[1.4. Vocabulary 4](#_Toc160752297)

[Infrared (IR) Communication: 4](#_Toc160752298)

[NEC Protocol: 4](#_Toc160752299)

[MicroPython: 4](#_Toc160752300)

[Carrier Frequency: 4](#_Toc160752301)

[Modulation/Demodulation: 4](#_Toc160752302)

[Breadboard: 4](#_Toc160752303)

[1.5. Hardware Setup: 5](#_Toc160752304)

[TSAL 6400 IR Emitter : 5](#_Toc160752305)

[Breadboard Setup for IR Emitter: 5](#_Toc160752306)

[Adafruit IR Receiver Specifications: 6](#_Toc160752307)

[Breadboard Setup for IR Receiver: 7](#_Toc160752308)

[1.6. Software Setup for IR Transmittion 8](#_Toc160752309)

[Importing IR Tx Library from Peter Hinch 8](#_Toc160752310)

[Project Creation and Library Upload to Raspberry Pi Pico 9](#_Toc160752311)

[1.7. Software Setup for IR Reception 11](#_Toc160752312)

[Importing IR Rx Library from Peter Hinch 11](#_Toc160752313)

[Project Creation and Library Upload to Raspberry Pi Pico 13](#_Toc160752314)

[1.8. Software Implementation : Script Implementation for IR Signal Transmission 15](#_Toc160752315)

[1.9. Software Implementation : Script Implementation for IR Signal Reception 16](#_Toc160752316)

[1.10. Running and Verifying IR Signal Reception and Transmission 17](#_Toc160752317)

[Uploading the Scripts to Raspberry Pi Pico: 17](#_Toc160752318)

[Running the Scripts: 17](#_Toc160752319)

[Testing the Transmission and Reception: 18](#_Toc160752320)

[Expected Outcome: 18](#_Toc160752321)

[1.11. Lab Deliverables 18](#_Toc160752322)

[IR Receiver and Transmitter Circuit Documentation:: 18](#_Toc160752323)

[Circuit Measurement Documentation: 19](#_Toc160752324)

[Script Execution and Output: 19](#_Toc160752325)

[Reflection and Troubleshooting: 19](#_Toc160752326)

[1.12. Additional Resources 19](#_Toc160752327)

## Equipment’s & Materials

### Raspberry Pi Pico

The primary microcontroller board for this lab, equipped with an RP2040 chip, providing ample GPIO pins for interfacing with external components.

### IR Receiver Module (TSOP382 from Adafruit)

A specific model of IR receiver tuned to a 38kHz carrier frequency, suitable for decoding signals from most standard IR remotes.

IR Transmitter (TSAL6400)

A high-power, 940nm infrared LED for emitting IR signals. Ideal for use in transmitting applications due to its high intensity and narrow beam angle

### Breadboard

A selection of male-to-male, male-to-female, and female-to-female jumper wires for making connections between the Raspberry Pi Pico, the IR receiver, and the breadboard.

### Resistors

A few resistors, including a 10kΩ pull-up resistor for the IR receiver's output pin if necessary, and additional resistors for any other circuit elements like LEDs.

### LED

Light-emitting diodes to provide visual feedback when IR signals are received and decoded. Include at least one LED and a suitable current-limiting resistor (220Ω to 1kΩ)

### Jumper Wires

Conductive wires for making connections on a breadboard.

### USB Cable:

For connecting the Raspberry Pi Pico to a computer.

### Visual Studio Code

Software for writing and uploading MicroPython code to the Raspberry Pi Pico.

## Vocabulary

### Infrared (IR) Communication:

A form of wireless communication that uses infrared light waves to transmit data over short distances. Commonly used in remote controls, IR communication allows for the control of various devices without the need for physical connections.

### NEC Protocol:

A popular protocol for IR communication, particularly in remote control systems. It outlines a specific set of rules for encoding and transmitting data over infrared signals, including data structure, bit length, and timing for signals.

### MicroPython:

A streamlined version of the Python 3 programming language designed to run on microcontrollers and embedded systems. It enables programming microcontrollers in Python, making it accessible for rapid prototyping and development.

### Carrier Frequency:

In the context of IR communication, it refers to the frequency of the infrared light wave that carries the encoded information. The most common carrier frequency for IR remote controls is 38 kHz, allowing the receiver to filter out signals from other sources.

### Modulation/Demodulation:

The process by which data is encoded onto a carrier wave (modulation) for transmission and subsequently extracted from the carrier wave by the receiver (demodulation). In IR communication, modulation involves varying the pulse width of the IR signal to encode data.

### Breadboard:

A solderless device for prototyping electronic circuits. It allows for the insertion of components and jumper wires into connected sockets to build and test circuit designs without permanent soldering, facilitating easy adjustments and experimentation.

## Hardware Setup:

### TSAL 6400 IR Emitter :

TSAL6400 is an infrared, 940 nm emitting diode in GaAlAs multi quantum well (MQW) technology with high radiant power and high speed.

#### Pin out :

A close-up of a silver object

Description automatically generated

Cathode -

Anode +

### Breadboard Setup for IR Emitter:

Note: Use separate breadboards for the IR Transmission circuit and the IR Receiver circuit, as detailed in the following section. The transmission circuit should be assembled on one breadboard and tested with a Raspberry Pi belonging to one of the team members. Meanwhile, the receiver circuit should be set up on another breadboard and programmed using a different laptop..

For the IR transmission side, we'll construct a circuit with the Raspberry Pi Pico to control an IR LED using an NPN transistor. This allows the Pico to drive the IR LED with adequate current without exceeding the GPIO pin's current limits.

#### Components:

* Raspberry Pi Pico
* NPN Transistor (e.g., 2N2222)
* IR Transmitter LED (TSAL6400)
* Current-limiting resistor for the IR LED
* Base resistor for the NPN transistor
* Jumper wires

#### Connection Steps:

* Connect GPIO Pin 17 of the Raspberry Pi Pico to the base of the NPN transistor through a suitable base resistor (1kΩ in our case).
* Connect the emitter of the NPN transistor directly to the ground (GND) on the Raspberry Pi Pico.
* Connect the collector of the NPN transistor to the cathode (negative side) of the IR LED through a suitable resistor (47Ω in our case).
* Connect the anode (positive side) of the IR LED to the VSYS pin of the Raspberry Pi. The VSYS pin provides 5 volts, which is suitable for powering the IR LED.
* Use VBUS or VSYS to power the breadboard and connect GND to ground the breadboard.

#### Note on Power Supply:

The VSYS pin on the Raspberry Pi Pico can be used to supply 5 volts to components that require it. It is important to note that when using VSYS to power external components, ensure that you do not draw more current than the Pico can provide. Always check the Pico's specifications for the maximum current draw. For the breadboard power supply, use the VBUS (5V from USB) or VSYS to power the components on the breadboard and connect a GND pin to the breadboard to complete the circuit's ground reference

A diagram of a circuit board

Description automatically generated

### Adafruit IR Receiver Specifications:

The Adafruit IR Receiver ( Adafruit model 1528-157-ND) is designed for easy integration into projects involving IR signal reception. It's a 38kHz IR receiver module suitable for decoding signals from most remote controls.

#### Pinout :

VCC (Power): Connect to 3.3V on the Raspberry Pi Pico.

GND (Ground): Connect to one of the ground pins on the Pico.

OUT (Signal Output): Connect to a GPIO pin (GPIO16) on the Pico for reading the IR signal.



OUT

GND

Vs

Figure 1 : Adafruit IR Pinout

A table of electrical applications

Description automatically generated with medium confidence

### Breadboard Setup for IR Receiver:

To facilitate an easy and reversible setup, we'll use a breadboard to connect the IR receiver to the Raspberry Pi Pico. This method allows for quick adjustments and troubleshooting. The receiver side of the lab is designed to receive four distinct commands from the IR transmitter. Upon successful reception of each command, a different LED (LED1 to LED4) will blink.

#### Components:

* Raspberry Pi Pico
* Adafruit IR Receiver
* LEDs
* 64 Ohm Resitors
* Breadboard
* Jumper wires

#### Connection Steps:

* Place the IR Receiver on the Breadboard: Carefully insert the three pins of the IR receiver into three separate rows on the breadboard. Ensure there's no short circuit between the pins.
* Power Connection (VCC): Use a jumper wire to connect the VCC pin of the IR receiver to one of the 3.3V pins on the Raspberry Pi Pico. This supplies power to the IR receiver.
* Ground Connection (GND): Use another jumper wire to connect the GND pin of the IR receiver to one of the grounds (GND) pins on the Pico. It's crucial to establish a common ground for all components.
* Signal Connection (OUT): Finally, use a jumper wire to connect the OUT pin of the IR receiver to a chosen GPIO pin on the Pico (e.g., GPIO15). This pin will be used to read the IR signals.
* Connect GPIO Pins 18 to 21 of the Raspberry Pi Pico to the anodes (positive sides) of LED1 to LED4, respectively, through a 64 Ohm resistor each.
* Connect the cathodes (negative sides) of LED1 to LED4 directly to the ground (GND) on the Raspberry Pi Pico.

A diagram of a circuit board

Description automatically generated

## Software Setup for IR Transmittion

### Importing IR Tx Library from Peter Hinch

To enable the Raspberry Pi Pico to send infrared signal, we'll utilize a dedicated library that simplifies the IR communication process. This section will guide you through the steps of downloading, installing, and utilizing the library for your project.

* Downloading the Library
* Visit the GitHub repository for the micropython-ir library: micropython-ir on GitHub by Peter Hinch.

Link : <https://github.com/peterhinch/micropython_ir/tree/master>

* Locate the "Code" button and click on it to reveal the dropdown menu.

A screenshot of a computer

Description automatically generated

* Select "Download ZIP" to download the library files to your computer.

### Project Creation and Library Upload to Raspberry Pi Pico

Note: Use two distinct laptops. Set up the Transmission project on one laptop and the Reception project on another.

Follow these steps to create a new project in Visual Studio Code for Lab 13 and upload the required IR receiver library to your Raspberry Pi Pico.

Step-by-Step Instructions:

* Open Visual Studio Code:
  + Launch Visual Studio Code on your computer.
* Create a New Project:
  + Navigate to the 'File' menu and select 'New Folder'. This will be your project directory.
  + Name the folder IR\_TX or a name of your choosing that indicates it's for Lab 13.
* Set Up the Project for Raspberry Pi Pico:
  + In Visual Studio Code, open the new folder you just created.
  + Configure the folder as a Pico project.
  + Your Explorer pane in Visual Studio Code should resemble the figure provided below, showing the project directory with the .vscode and micropico files.

A screenshot of a computer

Description automatically generated

Figure 2 : Project window initially

* Copy the IR Receiver Library Folder:
  + Copy the ir\_tx folder from the extracted micropython-ir library that you previously downloaded.
  + Paste the ir\_tx folder into your IR\_TX project directory.

A screenshot of a computer

Description automatically generated

Figure 3 : Project window with ir\_tx folder from library

* Upload the Library to Raspberry Pi Pico:
  + Connect your Raspberry Pi Pico to your computer via a USB cable.
  + In Visual Studio Code, right-click the project folder in the Explorer pane and select ‘Ctrl+Shift+P’ and type ‘>MicroPico:Upload project to Pico', as illustrated in the figure.

A screenshot of a computer

Description automatically generated

Figure 4 : Upload project to Pico

* + Wait for the upload to complete. A notification should appear indicating that the project has been uploaded successfully.

A screen shot of a black box

Description automatically generated

Figure 5 : Project uploaded to Raspberry

* Create a New Python Script:
  + Within the same project directory, create a new file by right-clicking and selecting 'New File'. Name this file as main.py.
* Import the Library in Your Script:
  + At the beginning of your Python script, add the following import statement to include the IR receiver library:

“import ir\_tx”

* Test The Library
  + Run your script by pressing the run button in Visual Studio Code.
  + If no errors occur, it confirms that the ir\_tx library is successfully imported and ready for use in your project. You can now proceed to utilize the library's functionality to develop your IR receiver application.

By following these steps, you ensure that your development environment is correctly set up and that the Raspberry Pi Pico is ready to receive and decode IR signals with the help of the uploaded library.

## Software Setup for IR Reception

**Note**: Use two different Laptops. Create a Transmission project in one laptop. Create Reception project in another.

### Importing IR Rx Library from Peter Hinch

To enable the Raspberry Pi Pico to decode infrared signals from remote controls, we'll utilize a dedicated library that simplifies the IR communication process. This section will guide you through the steps of downloading, installing, and utilizing the library for your project.

* Downloading the Library
* Visit the GitHub repository for the micropython-ir library: micropython-ir on GitHub by Peter Hinch.

Link : <https://github.com/peterhinch/micropython_ir/tree/master>

* Locate the "Code" button and click on it to reveal the dropdown menu.

*Extracting the Library Files*

* Navigate to the downloaded ZIP file on your computer.
* Right-click on the ZIP file and select "Extract All..." or use your preferred archive manager to extract the files.
* After extraction, you will see a folder named micropython\_ir-master containing various subfolders and files.

A screenshot of a computer

Description automatically generated

* Select "Download ZIP" to download the library files to your computer.

A screenshot of a computer

Description automatically generated

*Extracting the Library Files*

* Navigate to the downloaded ZIP file on your computer.
* Right-click on the ZIP file and select "Extract All..." or use your preferred archive manager to extract the files.
* After extraction, you will see a folder named micropython\_ir-master containing various subfolders and files.

### Project Creation and Library Upload to Raspberry Pi Pico

Follow these steps to create a new project in Visual Studio Code for Lab 13 and upload the required IR receiver library to your Raspberry Pi Pico.

Step-by-Step Instructions:

* Open Visual Studio Code:
  + Launch Visual Studio Code on your computer.
* Create a New Project:
  + Navigate to the 'File' menu and select 'New Folder'. This will be your project directory.
  + Name the folder IR\_RX or a name of your choosing that indicates it's for Lab 13.
* Set Up the Project for Raspberry Pi Pico:
  + In Visual Studio Code, open the new folder you just created.
  + Configure the folder as a Pico project.
  + Your Explorer pane in Visual Studio Code should resemble the figure provided below, showing the project directory with the .vscode and micropico files.

A black rectangle with white text

Description automatically generated

Figure 2 : Project window initially

* Copy the IR Receiver Library Folder:
  + Copy the ir\_rx folder from the extracted micropython-ir library that you previously downloaded.
  + Paste the ir\_rx folder into your IR\_RX project directory.

A screenshot of a computer

Description automatically generated

Figure 3 : Project window with ir\_tx folder from library

* Upload the Library to Raspberry Pi Pico:
  + Connect your Raspberry Pi Pico to your computer via a USB cable.
  + In Visual Studio Code, right-click the project folder in the Explorer pane and select ‘Ctrl+Shift+P’ and type ‘>MicroPico:Upload project to Pico', as illustrated in the figure.

A screenshot of a computer

Description automatically generated

Figure 4 : Upload project to Pico

* + Wait for the upload to complete. A notification should appear indicating that the project has been uploaded successfully.

A screen shot of a black box

Description automatically generated

Figure 5 : Project uploaded to Raspberry

* Create a New Python Script:
  + Within the same project directory, create a new file by right-clicking and selecting 'New File'. Name this file as main.py.
* Import the Library in Your Script:
  + At the beginning of your Python script, add the following import statement to include the IR receiver library:

“import ir\_rx”

A screenshot of a computer

Description automatically generated

Figure 6 : Importing using script

* Test The Library
  + Run your script by pressing the run button in Visual Studio Code.
  + If no errors occur, it confirms that the ir\_rx library is successfully imported and ready for use in your project. You can now proceed to utilize the library's functionality to develop your IR receiver application.

By following these steps, you ensure that your development environment is correctly set up and that the Raspberry Pi Pico is ready to receive and decode IR signals with the help of the uploaded library.

## Software Implementation : Script Implementation for IR Signal Transmission

Now that you have set up your Raspberry Pi Pico for IR transmission, use this script to send commands to the IR receiver. Make sure you have connected your IR LED to the Raspberry Pi Pico as detailed in the hardware setup instructions.

Steps for Implementing the IR Transmission Script:

* Open Your Project Folder in Visual Studio Code:
* Navigate to the directory for the IR transmission project you have created in Visual Studio Code.
* Create a New MicroPython Script for Transmission named main.py:
  + Right-click within the Explorer pane and select 'New File'.
  + Name the file main.py.
  + Copy the Transmission Code with Comments:

from machine import Pin

import uasyncio as asyncio

from ir\_tx.nec import NEC

# Define an asynchronous function to handle IR transmission

async def transmit\_ir():

ir\_transmitter = NEC(Pin(17, Pin.OUT, value=0)) # Initialize IR transmitter on Pin 17

addr = 0x01 # Example device address

commands = [0x01, 0x02, 0x03, 0x04] # List of commands to send

while True:

for command in commands:

ir\_transmitter.transmit(addr, command) # Send each command

print(f"IR signal transmitted: Addr {addr}, Command {command}")

await asyncio.sleep(3) # Wait for 3 seconds before sending the next command

# Main function to run the transmitter

async def main():

await transmit\_ir() # Call the transmit function

if \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main()) # Start the asynchronous event loop

* Create a New MicroPython Script for Transmission named main.py:
  + Right-click within the Explorer pane and select 'New File'.
  + Name the file main.py.
  + Copy the Transmission Code with Comments:
* Save the script by clicking 'File' > 'Save' or pressing Ctrl + S.

## Software Implementation : Script Implementation for IR Signal Reception

Once the ir\_rx library has been successfully uploaded to your Raspberry Pi Pico, you're ready to start receiving and decoding IR signals. The following script uses the NEC protocol to interpret signals from a standard IR remote control.

Steps for Implementing the Script:

* Open Your Project Folder in Visual Studio Code:
  + Navigate to the IR reception directory previously set up in Visual Studio Code.
* Create a New MicroPython Script:
  + Right-click within the Explorer pane and select 'New File'.
  + Name the file something indicative of its function, like main.py.
* Copy the following Code:

from machine import Pin

import uasyncio as asyncio

from ir\_rx.nec import NEC\_8 # Use the NEC 8-bit class

# Initialize GPIO pins for LEDs with corresponding command codes

led\_pins = {

0x01: Pin(18, Pin.OUT), # Command 0x01 associated with LED on Pin 18

0x02: Pin(19, Pin.OUT), # Command 0x02 associated with LED on Pin 19

0x03: Pin(20, Pin.OUT), # Command 0x03 associated with LED on Pin 20

0x04: Pin(21, Pin.OUT), # Command 0x04 associated with LED on Pin 21

}

# Asynchronous function to turn off LED after a delay

async def turn\_off\_led(led\_pin):

await asyncio.sleep(2) # Non-blocking wait for 2 seconds

led\_pin.value(0) # Turn off LED

# Callback function triggered when an IR code is received

async def ir\_callback(data, addr, \_):

print(f"Received NEC command! Data: 0x{data:02X}, Addr: 0x{addr:02X}")

if data in led\_pins: # If the command is one of the known commands

led\_pin = led\_pins[data] # Get the corresponding LED Pin object

led\_pin.value(1) # Turn on the LED

asyncio.create\_task(turn\_off\_led(led\_pin)) # Schedule turning off the LED

# Initialize and set up the IR receiver

ir\_pin = Pin(16, Pin.IN, Pin.PULL\_UP) # IR receiver on GPIO Pin 16

ir\_receiver = NEC\_8(ir\_pin, callback=ir\_callback) # Setup IR receiver with the callbackFunction `ir\_callback`.

# Optional, use print\_error for debugging if necessary

ir\_receiver.error\_function(print\_error)

# Asynchronous main loop to keep the script running

async def main():

while True:

await asyncio.sleep\_ms(100) # Sleep for 100 ms

# Run the asynchronous event loop

if \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

* Paste the Code into Your New Script:
  + Open the ir\_receiver\_script.py file in Visual Studio Code and paste the copied code.
* Adjust the GPIO Pin Number if Necessary:
  + If you've connected the IR receiver to a pin other than GPIOmentioned in the code, update the ir\_pin line with the correct GPIO number.
* Save the Script:
  + Save the script by clicking 'File' > 'Save' or by pressing Ctrl + S on your keyboard.

## Running and Verifying IR Signal Reception and Transmission

Once you have programmed both Raspberry Pi Picos with their respective IR transmission (IR\_TX) and IR reception (IR\_RX) scripts, it's time to test the complete setup. This process will allow you to verify the correct functionality of both the transmission and reception of IR signals..

### Uploading the Scripts to Raspberry Pi Pico:

* Uploading IR\_TX Script:
  + Open Visual Studio Code and navigate to the project directory containing the main.py for IR transmission.
  + Connect your Raspberry Pi Pico (designated as the transmitter) to your computer via USB.
  + Access the command palette with Ctrl+Shift+P (or Cmd+Shift+P on macOS) and type >MicroPico: Upload project to Raspberry Pi Pico. This will upload the main.py script to the transmitter Pico.
* Uploading IR\_RX Script:
  + Repeat the process for the Raspberry Pi Pico designated as the receiver, ensuring that its main.py script is the one used for IR reception

### Running the Scripts:

* Transmitter Pico:
  + Once the IR\_TX script is uploaded, as soon as the transmitter Pico is powered up, it will start sending the NEC messages.
* Receiver Pico:
  + Similarly, once the IR\_RX script is uploaded and the receiver Pico is powered, it will automatically start listening for NEC signals.

### Testing the Transmission and Reception:

* Verification Process:
  + Upon successful transmission, the receiver Pico should detect the NEC signals and the corresponding LEDs should blink according to the transmitted IR NEC codes.
* Terminal Output:
  + Observe the terminal output in Visual Studio Code. You should see printed messages indicating that NEC commands have been received with the data and address values from the transmitted IR signal.
* Troubleshooting:
  + It's common to encounter some missed or unknown frames due to the presence of other IR signals in the environment.
  + Ensure the main four codes sent by the transmitter are consistently being received by the receiver.
  + If the expected codes are not received, check for errors in the terminal, review the hardware connections, and confirm the proper alignment of the IR transmitter and receiver.

### Expected Outcome:

* Successful Reception:
  + A successful test is indicated by the receiver Pico consistently identifying the main four codes transmitted by the transmitter Pico and blinking the corresponding LEDs.
* Handling Ambient IR Signals:
  + Some interference from ambient IR signals is expected. However, the system should reliably recognize and respond to the signals transmitted by the IR\_TX setup.

By following these steps, you should be able to conduct a thorough test of the IR communication system you have built using the Raspberry Pi Picos. The hands-on experience will reinforce your understanding of the principles of IR signal transmission and reception.

## Lab Deliverables

For this lab, students will document the process and results of building and testing an IR communication system with both a transmitter and a receiver using the Raspberry Pi Pico.

### IR Receiver and Transmitter Circuit Documentation:

* Circuit Images:
  + Provide clear, well-lit photographs or neatly drawn schematics of both the completed IR transmitter and receiver circuits.
  + The Raspberry Pi Pico and all connections for each circuit, including power, ground, signal wires, and LEDs, should be visible and labeled.
* Circuit Descriptions:
  + Include a brief description for each image, outlining the purpose and function of each component within both the transmitter and receiver circuits.

### Circuit Measurement Documentation:

* Multimeter Readings:
  + Record and document the forward voltage across the IR LED and IR receiver when in operation.
  + Measure and record the current flowing through both the IR transmitter and receiver circuits.
* Analysis:
  + Compare the measurements against the datasheet specifications.
  + Analyze any discrepancies and discuss potential reasons and impacts on circuit functionality

### Script Execution and Output:

* Script Code:
  + A copy of the final MicroPython script used for IR signal transmission and reception .
  + Commented code explaining the function of critical lines or blocks of code.
* Terminal Output:
  + Screenshots or copied text from the Visual Studio Code terminal displaying the output of the script when IR signals are transmitted and received.

### Reflection and Troubleshooting:

* Lab Reflection:
  + A brief write-up reflecting on the lab experience, what was learned, and the practical skills gained.
* Troubleshooting Steps:
  + An account of any issues encountered during the lab and how they were resolved.
  + This can include hardware setup challenges, software bugs, or understanding the datasheet.

**Submission Format:**

* Compile all deliverables into a single PDF document, clearly sectioned and labeled.
* Ensure all images and screenshots are legible with appropriate captions or annotations.
* Students must submit their lab report by the deadline specified by the instructor. Reports should demonstrate a clear understanding of the lab's objectives and outcomes, reflecting the student's engagement with both the practical and theoretical aspects of IR communication.

## Additional Resources

Adafruit 1528-157-ND IR receiver datasheet

<https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/5389/157_Web.pdf>

TSAL 6400 IR Emitter

https://www.vishay.com/docs/81011/tsal6400.pdf

NEC IR protocol specification

<https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>

NEC Infrared receiver class Micropython

<https://forum.micropython.org/viewtopic.php?t=671>