# Lab 13 – SBB IR Receiver & programming

## Lab Overview

This lab introduces students to the practical aspects of infrared (IR) communication through hands-on experience with the Raspberry Pi Pico and an IR receiver, utilizing the Adafruit TSOP382. By incorporating the [MicroPython](https://github.com/peterhinch/micropython_ir/tree/master) library developed by Peter Hinch, students will explore how to set up a circuit for IR signal reception, decode NEC protocol frames, and apply this knowledge to control actions based on IR commands. The goal is to provide a foundational understanding of IR communication systems and their application in real-world scenarios, bridging the gap between theoretical knowledge and practical implementation.

## Learning Outcomes

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

* Understand IR Communication: Grasp the basic principles behind IR communication, including signal modulation and demodulation.
* Set Up an IR Receiver: Assemble an IR receiver circuit with the Raspberry Pi Pico and configure it for signal detection.
* Decode IR Signals: Utilize programming techniques to interpret IR signals and translate them into actionable commands.
* 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](#_Toc160531996)

[1.1. Lab Overview 1](#_Toc160531997)

[1.2. Learning Outcomes 1](#_Toc160531998)

[1.3. Equipment’s & Materials 3](#_Toc160531999)

[Raspberry Pi Pico 3](#_Toc160532000)

[IR Receiver Module (TSOP382 from Adafruit) 3](#_Toc160532001)

[Breadboard 3](#_Toc160532002)

[Resistors 3](#_Toc160532003)

[LED 3](#_Toc160532004)

[Jumper Wires 3](#_Toc160532005)

[USB Cable: 3](#_Toc160532006)

[Visual Studio Code 3](#_Toc160532007)

[1.4. Vocabulary 3](#_Toc160532008)

[Infrared (IR) Communication: 3](#_Toc160532009)

[NEC Protocol: 3](#_Toc160532010)

[MicroPython: 4](#_Toc160532011)

[Carrier Frequency: 4](#_Toc160532012)

[Modulation/Demodulation: 4](#_Toc160532013)

[Breadboard: 4](#_Toc160532014)

[1.5. Hardware Setup: 5](#_Toc160532015)

[Adafruit IR Receiver Specifications: 5](#_Toc160532016)

[Breadboard Setup for IR Receiver: 5](#_Toc160532017)

[1.6. Software Setup 6](#_Toc160532018)

[Importing IR Rx Library from Peter Hinch 6](#_Toc160532019)

[Project Creation and Library Upload to Raspberry Pi Pico 7](#_Toc160532020)

[1.7. Software Implementation : Script Implementation for IR Signal Reception 9](#_Toc160532021)

[Run the Script: 10](#_Toc160532022)

[Test with an IR Transmitter: 11](#_Toc160532023)

[1.8. Lab Deliverables 11](#_Toc160532024)

[1. IR Receiver Circuit Documentation: 11](#_Toc160532025)

[2. Data Sheet Analysis: 11](#_Toc160532026)

[3. Circuit Measurement Documentation: 12](#_Toc160532027)

[4. Script Execution and Output: 12](#_Toc160532028)

[5. Reflection and Troubleshooting: 12](#_Toc160532029)

[1.9. Additional Resources 12](#_Toc160532030)

## 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.

### 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:

### 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 (GPIO15) 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.

#### Components:

* Raspberry Pi Pico
* Adafruit IR Receiver
* 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.

## Software Setup

### 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.

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 LAB13\_PART1 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 LAB13\_PART1 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 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 LAB13\_PART1 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 ir\_receiver\_script.py.
* Copy the following Code:

from machine import Pin

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

from ir\_rx.print\_error import print\_error # for debugging

# Callback function to execute when an IR code is received

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

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

# Setup the IR receiver

ir\_pin = Pin(16, Pin.IN, Pin.PULL\_UP) # Adjust the pin number based on your wiring

ir\_receiver = NEC\_8(ir\_pin, callback=ir\_callback)

# Optional: Use the print\_error function for debugging

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

# Main loop to keep the script running

while True:

pass # Execution is interrupt-driven, so just keep the script alive

* 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 GPIO 16, 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

After setting up your Raspberry Pi Pico with the IR receiver library, it's time to test the reception with the assistance of your Teaching Assistant (TA). Here's how you can run your script and verify the reception of IR signals.

### Run the Script:

* Execute the Script:
  + With the ir\_receiver\_script.py file open in Visual Studio Code, start the script by clicking on the 'Run' button located at the top-right of the editor or by pressing F5.
  + Alternatively, you can use the command palette with the shortcut Ctrl+Shift+P (or Cmd+Shift+P on macOS), type 'Run Python File', and hit Enter.

### Test with an IR Transmitter:

* Prepare for Testing:
  + Ensure your Raspberry Pi Pico is connected and set up according to the previous steps, with the ir\_rx library uploaded and the script ready to run.
* TA Assistance for Signal Transmission:
  + Your TA will use a standard IR transmitter to send various NEC commands toward your receiver setup.
  + Align the transmitter's IR LED with your Pico's IR receiver module for clear signal transmission.
* Observe Terminal Output:
  + Watch the output in Visual Studio Code's integrated terminal.
  + A successful reception will display messages like "Received NEC command!" followed by specific data and address values from the received IR signal.
* Troubleshoot if Necessary:
  + If the expected output isn't appearing, review the terminal for any error messages that might provide clues for debugging.
  + Double-check your hardware connections and ensure your IR receiver is correctly positioned to receive signals from the transmitter.

By following these steps, you'll not only run your script but also engage in a practical test of the IR reception capabilities of your Raspberry Pi Pico, giving you hands-on experience with IR communication protocols.

## Lab Deliverables

For Lab 13, students are expected to provide evidence of their work and understanding of IR communication using a Raspberry Pi Pico and an IR receiver. The following are the detailed deliverables each student must submit:

### 1. IR Receiver Circuit Documentation:

* **Circuit Image:**
  + A clear, well-lit photograph or a neatly drawn schematic of the completed IR receiver circuit.
  + The Raspberry Pi Pico and all connections, including power, ground, and signal wires to the IR receiver, should be visible and labeled.
* **Circuit Description:**
  + A brief description accompanying the image, explaining the purpose of each component within the circuit.

### 2. Data Sheet Analysis:

* **Forward Voltage and Current:**
  + A written summary or a table extracted from the IR receiver's datasheet detailing the forward voltage (Vf) and the typical forward current (If) specifications.
  + An explanation of how these values relate to the functionality of the IR receiver in the context of the lab.

### 3. Circuit Measurement Documentation:

* **Multimeter Readings:**
  + Document the measured forward voltage across the IR receiver when exposed to an IR signal.
  + Record the current flowing through the IR receiver circuit when activated.
* **Analysis:**
  + Compare the measured values against the datasheet specifications.
  + Discuss any discrepancies and their possible causes or implications on the circuit's performance.

### 4. Script Execution and Output:

* **Script Code:**
  + A copy of the final MicroPython script used for IR signal 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 received.

### 5. 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>

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>

#include <IRremote.h>

const int irSendPin = 3; // The digital pin connected to the transistor's base

IRsend irsend(irSendPin); // Initialize the IRsend object

void setup() {

// If your version of IRremote does not support specifying the pin, you might need to modify the library or use the default pin.

}

void loop() {

irsend.sendNEC(0xFFA25D, 32); // Send the first NEC code

delay(2000); // Wait for 2 seconds before sending the next command

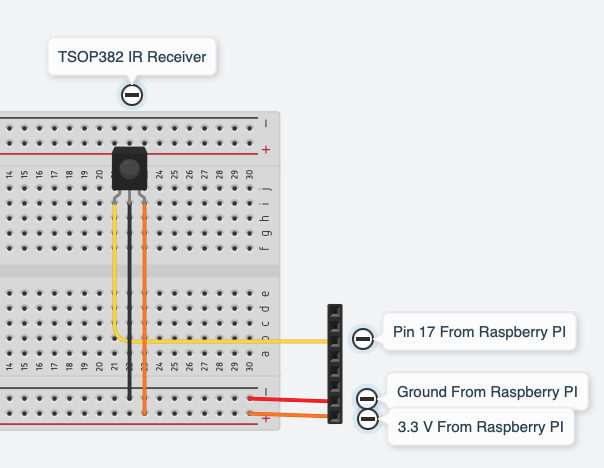
irsend.sendNEC(0xFF629D, 32); // Send the second NEC code

delay(3000); // Wait for 3 seconds before sending the next command

irsend.sendNEC(0xFFE21D, 32); // Send the third NEC code

delay(4000); // Wait for 4 seconds before starting the loop over again

}



A computer screen shot of a computer

Description automatically generated