# Micro:bit Morse Code Communication Documentation

Documentation created by Huzaifaa Hassan Latheef, UWE Student number: 22040156

## Introduction

Welcome to the Micro:bit Morse Code Communication project documentation. This project introduces embedded development with the BBC Micro:bit and demonstrates a simple communication protocol using Morse code between two Micro:bit devices. The goal is to provide hands-on experience in programming, IoT concepts, and basic communication protocols.

## How to Set Up and Use the Micro:bit Morse Code Communication

### Sender Micro:bit

1. Load the provided Python code onto the sender Micro:bit using the Micro:bit Python editor.
2. Power on the sender Micro:bit.
3. Press button A to send a dot ('.') signal or button B to send a dash ('-') signal in Morse code.
4. The LED matrix will display the corresponding signal (dot or dash) as you press the buttons.

### Receiver Micro:bit

1. Load the provided Python code onto the receiver Micro:bit using the Micro:bit Python editor.
2. Power on both the sender and receiver Micro:bits.
3. Ensure that both Micro:bits are set to the same radio group (e.g., group 6).
4. The sender Micro:bit will display signals on its LED matrix as you press buttons A and B.
5. The receiver Micro:bit will decode the received Morse code signals, apply a cipher, and display the corresponding characters.

## How It Works

The Micro:bit Morse Code Communication project uses a basic communication protocol based on Morse code. Each Micro:bit acts as a sender and a receiver. Pressing button A on the sender Micro:bit sends a dot ('.') signal, while pressing button B sends a dash ('-') signal. These signals are transmitted using radio communication. The receiver Micro:bit decodes the Morse code signals, applies a simple substitution cipher, and displays the resulting characters on its LED matrix.

## Cipher Explanation

In the Micro:bit Morse Code Communication project, a cipher is applied to the decoded Morse code signals to enhance the communication protocol. The cipher is a substitution cipher that replaces each decoded Morse code character with an alternative character. This substitution adds an element of encryption and complexity to the transmitted messages.

The cipher mapping is defined using the cipher\_map dictionary in the receiver's Python code. Each character is mapped to a corresponding replacement character. For example, the letter "A" is replaced with "Q," "B" with "R," and so on. Similarly, numbers are also replaced with alternative numbers in the cipher.

When the receiver Micro:bit decodes the Morse code signals and translates them using the morse\_code\_map, it then applies the cipher to obtain the actual characters displayed on the LED matrix. This simple cipher adds an extra layer of intrigue and fun to the project's communication process.

## Code

**Senders code:**  
from microbit import \*

import radio

radio.config(group=6)

radio.on()

def send\_signal(signal):

radio.send(signal)

sleep(200) # Adjust the interval if needed

while True:

if button\_a.was\_pressed():

display.show('.')

send\_signal('.')

display.clear()

elif button\_b.was\_pressed():

display.show('-')

send\_signal('-')

display.clear()

**Receivers Code:**

from microbit import \*

import radio

radio.config(group=6)

radio.on()

morse\_code\_map = {

".-": "A", "-...": "B", "-.-.": "C", "-..": "D", ".": "E", "..-.": "F", "--.": "G",

"....": "H", "..": "I", ".---": "J", "-.-": "K", ".-..": "L", "--": "M", "-.": "N",

"---": "O", ".--.": "P", "--.-": "Q", ".-.": "R", "...": "S", "-": "T", "..-": "U",

"...-": "V", ".--": "W", "-..-": "X", "-.--": "Y", "--..": "Z",

".----": "1", "..---": "2", "...--": "3", "....-": "4", ".....": "5",

"-....": "6", "--...": "7", "---..": "8", "----.": "9", "-----": "0"

}

# Define your substitution cipher mapping

cipher\_map = {

"A": "Q", "B": "R", "C": "S", "D": "T", "E": "U", "F": "V", "G": "W",

"H": "X", "I": "Y", "J": "Z", "K": "A", "L": "B", "M": "C", "N": "D",

"O": "E", "P": "F", "Q": "G", "R": "H", "S": "I", "T": "J", "U": "K",

"V": "L", "W": "M", "X": "N", "Y": "O", "Z": "P",

"1": "9", "2": "8", "3": "7", "4": "6", "5": "5",

"6": "4", "7": "3", "8": "2", "9": "1", "0": "0"

}

def decode\_morse(morse\_word):

decoded\_word = ''.join([morse\_code\_map.get(code, '') for code in morse\_word.split(' ')])

decoded\_message = ''.join([cipher\_map.get(decoded\_word[i:i+2], '?') for i in range(0, len(decoded\_word), 2)])

return decoded\_message

def receive\_morse():

morse\_code = ''

while True:

signal = radio.receive()

if signal:

morse\_code += signal

else:

break

return morse\_code

while True:

morse\_signal = receive\_morse()

decoded\_message = decode\_morse(morse\_signal)

display.scroll(decoded\_message)

sleep(4000) # Display for 4 seconds, adjust as needed

display.clear()

## Conclusion

The Micro:bit Morse Code Communication project showcases the implementation of a basic communication protocol using Morse code and a simple substitution cipher. By interacting with sender and receiver Micro:bits, you gain insights into embedded systems, programming, and IoT concepts.