WORKSHEET 2

Internet of things

Ahmed Ali (S2201026)

UFCFVK-15-2

CONTENTS

Lets get into the code ;

```
from microbit import *
import radio
```

1. Import Libraries: The code starts by importing the necessary libraries: **microbit** and **radio**. The **microbit** library provides access to Microbit's features, including the LED matrix and buttons, while the **radio** library enables communication between multiple Microbits using radio signals.

```
ALPH_MORSE = {
    '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': '-----', '.': '.-.-.-', ',': '--..--', '?': '..--..',
    " ": " "
}

MORSE_ALPH = {value: key for key, value in ALPH_MORSE.items()}
```

2. Morse Code Dictionaries: Two dictionaries are defined: **ALPH_MORSE** and **MORSE_ALPH**. **ALPH_MORSE** maps characters (letters, numbers, and some

```
# Caesar Cipher Encryption function
def encrypt_caesar(plain_text, shift):
    encrypted_text = ""
    for char in plain_text:
        if char in ALPH_MORSE:
            encrypted_text += ALPH_MORSE[char] + ' '
    return encrypted_text

# Caesar Cipher Decryption function
def decrypt_caesar(encrypted_text, shift):
    words = encrypted_text.strip().split(' ')
    decrypted_message = ''
    for word in words:
        if word in MORSE_ALPH:
            decrypted_message += MORSE_ALPH[word]
        else:
            decrypted_message += ' '
    return decrypted_message
```

punctuation) to their respective Morse code representations. **MORSE_ALPH** is the reverse mapping, allowing us to convert Morse code back to characters.
3. Caesar Cipher Encryption Function: The **encrypt_caesar** function takes a **plain_text** string and a **shift** value as input. It iterates over each character in the **plain_text** and converts it to its Morse code representation using the **ALPH_MORSE** dictionary. The encrypted Morse code is built and returned as **encrypted_text**.

4. Caesar Cipher Decryption Function: The **decrypt_caesar** function takes an **encrypted_text** string and a **shift** value as input. It splits the Morse code into individual words and converts each word back to its character representation using the **MORSE_ALPH** dictionary. The decrypted message is built and returned as **decrypted_message**.

```
radio.on()
radio.config(channel=1)
radio.RATE_1MBIT
```

5. Setting Up Radio Communication: The code turns on the radio (**radio.on()**) and configures it to use channel 1 (**radio.config(channel=1)**). Both Microbits need to be set to the same channel to communicate with each other. The radio rate is set to 1Mbit (**radio.RATE_1MBIT**).

```
while True:
    message = "I AM AHMED".lower()

    if button_a.was_pressed():
        # Encrypt the message using Caesar Cipher (with a fixed shift of 3 for example)
        shift = 3
        encrypted_morse_code = encrypt_caesar(message, shift)
        display_text = "Sending: " + encrypted_morse_code
        display.scroll(display_text)

        radio.send(encrypted_morse_code)

    received = radio.receive()
    if received:
        # Decrypt the received message using Caesar Cipher with the same shift value
        shift = 3
        decrypted_message = decrypt_caesar(received, shift)
        message = "Received: " + decrypted_message
        display.scroll(message)

    sleep(1000)
    display.clear()
```

Main Loop: The code enters an infinite loop (**while True:**) to keep the Microbit running continuously.

```
while True:
    message = "I AM AHMED".lower()
```

Sending Messages: If button A is pressed (**if button_a.was_pressed():**), the code encrypts the message "I AM AHMED" (you can change this to any desired message) using the Caesar Cipher with a fixed shift of 3 (**shift = 3**). It then displays the encrypted Morse code on the LED matrix and sends it to the other Microbit using radio communication (**radio.send(encrypted_morse_code)**).

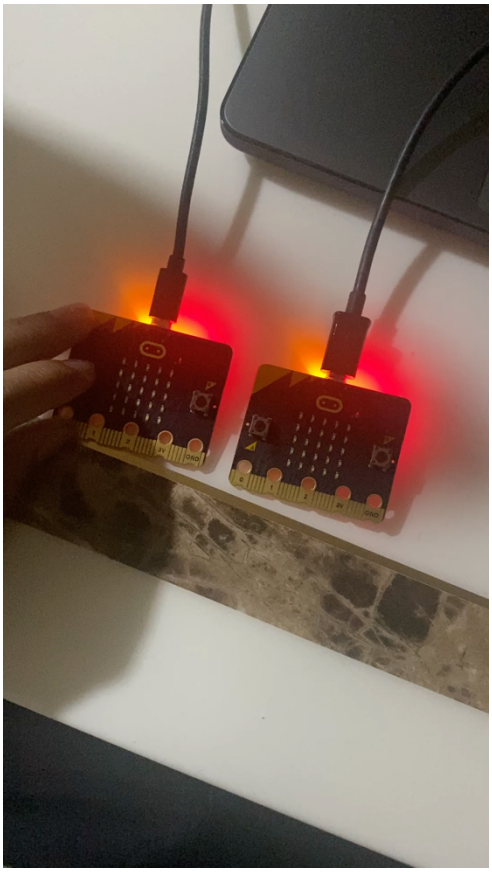
Receiving Messages: The code checks if there is any received radio signal (received = radio.receive()). If there is, it means the other Microbit sent an encrypted message. The code then decrypts the received Morse code using the same Caesar Cipher with a shift of 3, displays the decrypted message on the LED matrix, and scrolls it to be visible (display.scroll(message)).

Sleep and Clear: After each iteration of the loop, the code sleeps for 1000 milliseconds (**sleep(1000)**) to avoid unnecessary processing. The LED matrix is then cleared (**display.clear()**) before the next iteration.

Communication Between Microbits:

To make this code work on two Microbits, I have uploaded the exact same code to both devices. **Each Microbit will act as both a sender and a receiver.** When you press button A on one Microbit, it will send an encrypted message via radio to the other Microbit. The other Microbit will receive the message, decrypt it, and display the original message.

Here is a demonstration of the two device communication:



Python Code

```python
from microbit import *
import radio

ALPH_MORSE = {
    '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': '-----', '.': '.-.-.-', ',': '--..--', '?': '..--..',
    " ": " "
}

MORSE_ALPH = {value: key for key, value in ALPH_MORSE.items()}

# Caesar Cipher Encryption function
def encrypt_caesar(plain_text, shift):
    encrypted_text = ""
    for char in plain_text:
        if char in ALPH_MORSE:
            encrypted_text += ALPH_MORSE[char] + ' '
    return encrypted_text

# Caesar Cipher Decryption function
def decrypt_caesar(encrypted_text, shift):
    words = encrypted_text.strip().split(' ')
    decrypted_message = ''
    for word in words:
        if word in MORSE_ALPH:
            decrypted_message += MORSE_ALPH[word]
        else:
            decrypted_message += ' '
    return decrypted_message
display.show("!")

radio.on()
radio.config(channel=1)
radio.RATE_1MBIT

while True:
    message = "I AM AHMED".lower()

    if button_a.was_pressed():
        # Encrypt the message using Caesar Cipher (with a fixed shift of 3 for example)
        shift = 3
        encrypted_morse_code = encrypt_caesar(message, shift)
        display_text = "Sending: " + encrypted_morse_code
        display.scroll(display_text)

        radio.send(encrypted_morse_code)

    received = radio.receive()
    if received:
        # Decrypt the received message using Caesar Cipher with the same shift value
        shift = 3
        decrypted_message = decrypt_caesar(received, shift)
        message = "Received: " + decrypted_message
        display.scroll(message)

    sleep(1000)
    display.clear()
```