EL203
Project Report : Team 12
Morse Code Transmitter Decrypter

Group Members:

1. 201401070 - Anurag Gupta
2. 201401072 - N. Srisai Karthik
3. 201401073 - Pitta Nancy
4. 201401074 - Rashi Gupta
5. 201401075 - Deepit Patel

Aim:

Send encrypted messages from a source and decrypt the message at the receiving end using two Beaglebone Black.

Project Description:

- Previous Commitment Accomplished.

  At the source, Morse code key will be used to transmit signals where each letter has a specific frequency. The BeagleBone Black at the source end will generate the data for playing a tone. This data is sent through the receiver using socket programming over ethernet interface.

  At the receiver end, the dots and dashes are identified and they will be decoded into the original text which was to be transmitted.

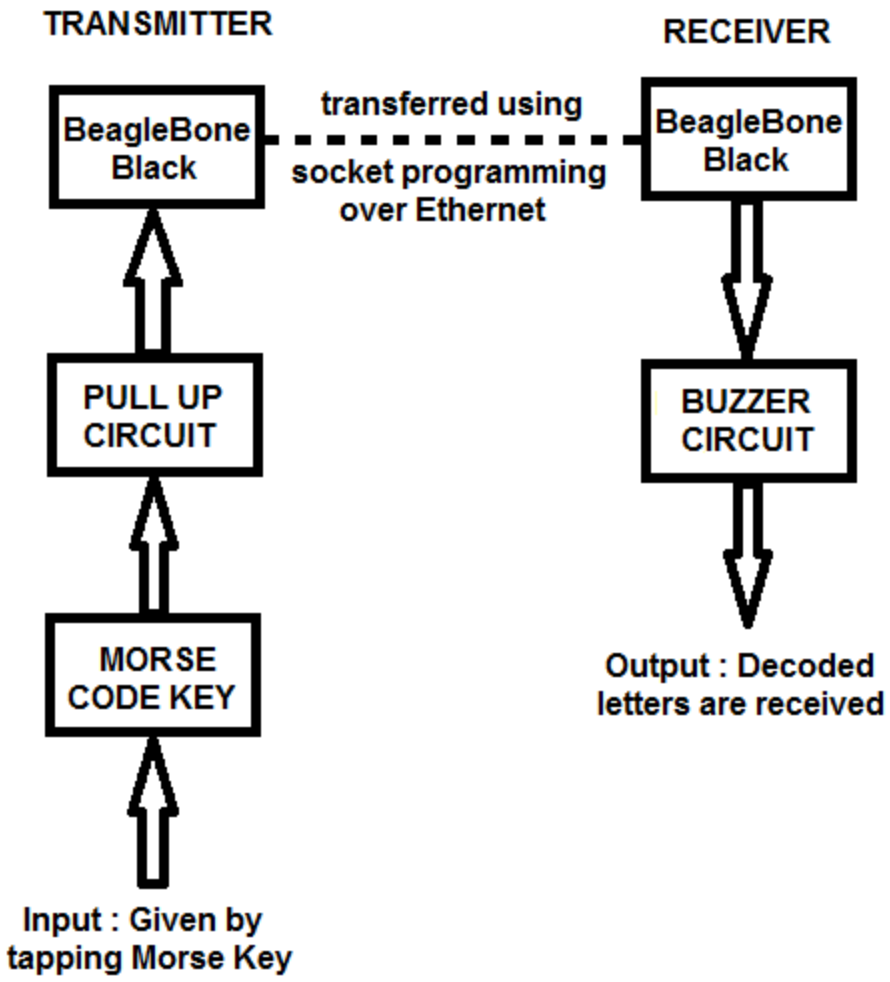
- Extension

  A particular task can be performed once certain specific words are received. This idea was extended and when the word “SOS” is received, the buzzer will start at the receiver end.

Materials Used:

1) BeagleBone Black x2
2) Morse code Key
3) Resistors
4) Jump wires
5) Buzzer

Block Diagram:



Implementation:

On the transmitter side, functions *wait_for_up()* and *wait_for_down()* use a while loop that makes the BeagleBone sleep until the pin state has changed. In our project, we have considered that a dot is about 0.15 seconds or less and anything longer than this is a dash.

The transmitter and the receiver are connected through ethernet and the data is transferred to the receiver using socket programming.

The expected behaviour is to print letter at the receiver end when someone from the transmitter end finish keying in a letter. Also when keying in word is finished, a space character will be displayed.

The concept of multithreading is used at the receiver end. As our main thread on the receiver end is always involved in receiving the data sent, we need to have another thread which can constantly do the work of decoding what the user is keying in from the transmitter end. So a new thread is required which will use the *decode()* function to see if the buffer contents matches a letter; it will also empty the buffer. If the gap of silence gets longer, this denotes a new word and a space character would be displayed. As soon as we receive signal SOS “...---...” (distress signal), again a new thread is called which activates the alarm at the receiver end.

Scope:

Morse Code is most prevalent in Aviation and Aeronautical fields since radio navigational aids such as VOR's and NDB's still identify in Morse Code. The US Navy and Coast Guard still use signal lamps to communicate via Morse Code.

Morse Code has also been used as an alternative form of communication for people with disabilities or who have their abilities to communicate impaired by stroke, heart attack, or paralysis. There have been several cases where individuals have been able to use their eyelids to communicate in Morse Code by using a series of long and quick blinks to represent that dots and dashes.

Transmitter Code:

```
import time
import Adafruit_BBIO.GPIO as GPIO
import socket

pin = "P9_42"
GPIO.setup(pin, GPIO.IN)

def wait_for_down(pin):
        while GPIO.input(pin):
                time.sleep(0.01)

def wait_for_up(pin):
        while not GPIO.input(pin):
                time.sleep(0.01)

down_time = 0
down_length = 0
up_time = 0

print("Ready")

s = socket.socket()
host = "10.100.70.192"
port = 43
s.bind((host, port))
s.listen(1000)

while True:
        wait_for_down(pin)
        down_time = time.time()     # record the time when the key was pressed
        wait_for_up(pin)
        up_time = time.time()       # record the time when the key was released
        down_length = up_time - down_time
        # get the length of time it was held down for
        c, addr = s.accept()
        msg = str(down_length)
```

```
        c.send(msg)
        print "Connected!"
        c.close()
        time.sleep(0.1)
```

Receiver Code:

```
import time
import Adafruit_BBIO.GPIO as GPIO
import thread
import socket
import sys

pin = "P9_42"
GPIO.setup(pin, GPIO.OUT)

morse_code = {
        ".-":     "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",
        "-.-.--":   "!"
}

def decode(bit_string):
        if bit_string in morse_code.keys():
                ch = morse_code[bit_string]
                sys.stdout.write(ch)
                if ch != ' ':
                        stack.append(ch)
                        if len(stack) >= 5: #1000:
                                stack.pop(0)
                if ''.join(stack) == "SOS!":
                        thread.start_new_thread(sosbeep, ())
                        del stack[:]
                sys.stdout.flush()

def sosbeep():
        GPIO.output(pin, GPIO.HIGH)
        time.sleep(3)
        GPIO.output(pin, GPIO.LOW)

def decoder_thread():
        global offset
        global stack
```

```
        global buffer
        new_word = False
        while True:
                time.sleep(.01)
                up_length = time.time() - offset
                if len(buffer) > 0 and up_length >= 1.5:
                        new_word = True
                        bit_string = "".join(buffer)
                        decode(bit_string)
                        del buffer[:]
                elif new_word and up_length >= 4.5:
                        new_word = False
                        sys.stdout.write(" ")
                        sys.stdout.flush()

DASH = '-'
DOT = '.'

down_length = 0
offset = 0
buffer = []
stack = []

thread.start_new_thread(decoder_thread, ())

while True:
        s = socket.socket()
        host = "10.100.70.192"
        port = 43
        s.connect((host,port))
        down_length = float(s.recv(1024))
        offset = time.time()
        buffer.append(DASH if down_length > 0.15 else DOT)
        s.close()
```