Raspberry Pi - Morse Code Reader
Project Plan Brief

Author: Yashiv Fakir

DATE: 8 JANUARY 2021

---

1 Overview and Objectives

Morse code is a method of communication that uses a combination of long and short signals/pulses to represent alphanumeric symbols, that are then combined and transmitted in various ways to formulate words/sentences. For more information on Morse code refer to [1] or from [2]

The primary objective of this project is to decode a Morse encoded message to plain text. The encoded message can be inputted as either:

1. a visual representation (dots and dashes black in colour) depicted on a sheet of white paper
2. using the long and short flashing of light emulated from a single LED
3. using two different coloured LED's to represent a dot and dash that flash at periodic intervals

The secondary objectives is then to determine an arrangement of sensors that are both cost effective and accurate to decipher the encoded input message. Then, utilise a Raspberry Pi to interface with the sensor arrangement to decipher the message.

2 Possible Designs

2.1 Design 1

2.1.1 Concept

Using an LDR and a light source, such as an LED, to shine on the paper encoded message mentioned in the previous section. The paper would have to be moved from under the sensor or the sensor would have to be moved along the paper at a constant rate. Then use measured time that the LDR detects a black part of the paper (when resistance is high) to differentiate from a dash (long time period) or a dot (short time period) or a space which is when white is detected (when resistance is low).

The same could be done when using an LED or LED's to transmit the message using the same parts. The LED that was used to emit the light into the paper, is set to be the source of encoded message that shines directly onto the LDR. Then using the time periods of the flashes to decode the message.

2.1.2 Part List

- 1x Raspberry Pi
- 1x LDR (Light Dependent Resistor)
- 1x ADC (Analog to Digital Converter)
- 2x resistors
- 1x LED
- 1x Breadboard
- Connector Leads (male-male and female-male)

2.1.3 Potential Problems

During the process of analysing the encoded input, other light sources in the external environment could affect the reading of the LDR sensor. The rate at which the input is read could vary leading to a false or incorrect reading from either an LED input or a paper input for the encoded message. Another possible problem could be the thickness of the dots and dashes drawn for the paper input that could potentially be too thin to be sensed by the LDR.

2.2 Design 2

2.2.1 Concept

Another possible design involves using a camera connected to a Raspberry Pi. There are several libraries that all ready exist that can aid in recognising patterns in images in real time. The best library that could help with this task is a library called OpenCV. OpenCV is cross platform on both C and Python with support. Then utilize the libraries capabilities as well as write your own code to convert the image from the camera to a digital encoded form of possibly 1's and 0's and decode the message.

There are several camera modules that are specifically configured for Raspberry Pi's however can be relatively expensive of approximately R300 - R500. There is a camera component that can interface with a Raspberry Pi Zero using the basic ports. This design will work better using a paper encoded message rather then an LED encoded message.

2.2.2 Part List

- 1x Raspberry Pi
- 1x LED (For Morse code input message)
- 1x Different coloured LED (Only necessary if the third Morse code input is used)
- 1x Breadboard
- 1x Camera Module

2.2.3 Potential Problems

The image processing software will have to account for any inconsistencies if utilising a paper drawn Morse coded message (lines or smudges that were left from a marker). The thickness of the dots and dashes drawn for the paper input could potentially be too thin to be sensed by the camera.
Then if utilizing LED's as an input, the camera software may find it difficult to differentiate between the background and the flash of the LED's. Also the camera would have to take images at a much faster rate than the rate of the flashing LED's to accurately differentiate the difference between and flash and dash which would also require lots of memory to be used.

3 Milestones

Note for this project Design 1 will be attempted first and then possibly Design 2 if Design 1 fails or is for the most part inefficient.

The specific tasks and schedule for this project is stipulated below:

1. Determine programming language to be used between Python and C. Currently leaning towards C for the entire project.
   Due: 12 January 2021

2. Determine the exact software libraries and functions that are to be used. Then summarize in a report.
   Due: 18 January 2021

3. Determine the exact hardware components and the layout on the breadboard and when interfacing with the Raspberry Pi that will be required for the project. Then summarize in a report.
   Due: 23 January 2021

4. Assemble the circuit and write the necessary code.
   Due: 6 February 2021

5. Code the 3 possible input cases. Test both the code and circuit with the 3 different inputs and make possible improvements.
   Due: 10 February 2021

6. Record all finds and outcomes of the project in a report.
   Due: 21 February 2021

4 Github Repository

All code and diagrams for this project can be sources at the link below:
https://github.com/yashivfakir/raspberryPI-morse_code_reader.git

References

[1] R. Burns, "Morse code," Dec 2020. [Online]. Available: https://en.wikipedia.org/wiki/Morse_code

[2] B. Editors, "Morse code." [Online]. Available: https://www.britannica.com/topic/Morse-Code