Project 1
Arduino Morse Code Reading

Project Description:

The purpose of this project is to create a sender and receiver system working with Morse code protocol. In this project, you should write a program to send the received string on the Serial Monitor console to the other Arduino Board using a GPIO (General Purpose Input Output) pin. We refer to the Morse code generator as ***Sender***.

The manual of all parts are available in the reference section of the Canvas.

Required Hardware:

- Two Arduino Mega 2560 and two cables.
- 1 Wire (to connect the digital I/O pins of both devices)
- Two Computers with USB port

Part 1

In this project, you have to first follow the Arduino online tutorial at the following address: https://www.arduino.cc/en/Hacking/LibraryTutorial in order to create a Morse code generation library that generates the Morse code as long and short flashes of the onboard led on pin 13. This project will familiarize you with the Arduino environment as well as their libraries.

Part 2

In this part, we would extend our library, created in part 1, to implement a converter that converts a string to Morse code. The Morse code protocol is shown in the following diagram:

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

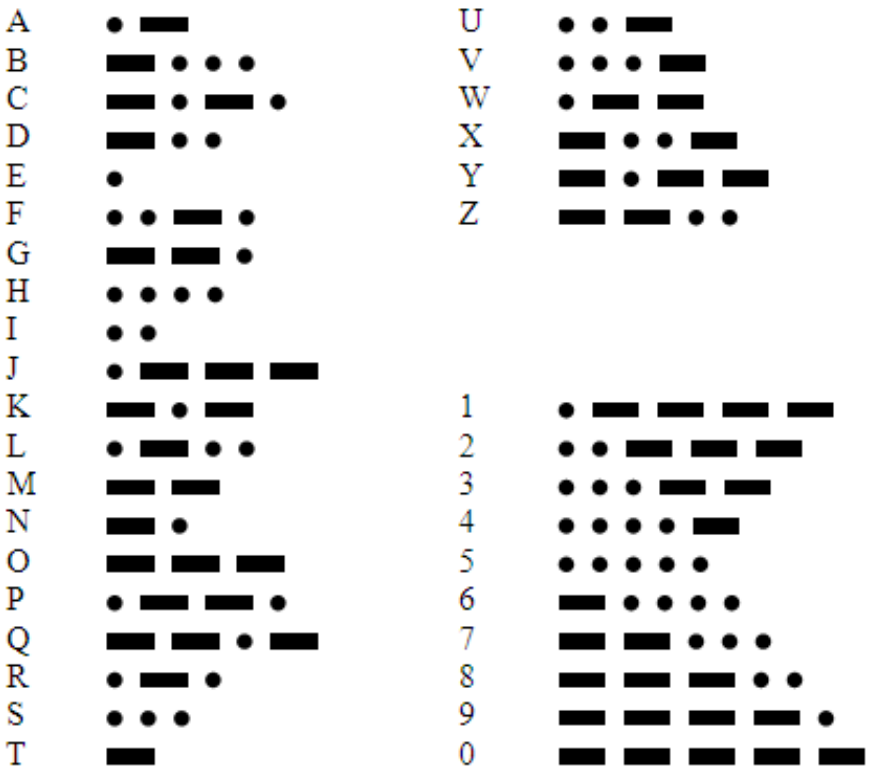

Figure 1: International Morse code

There are two terminologies in Morse code, dot and dash, which are used to represent the alphabets. For a “dot”, the LED should be stayed on for **250 milliseconds** and for a “dash”, the LED should be stayed on for **750 milliseconds**. The LED must be off between any two symbols (dot or dash) for **250 milliseconds**.

Between **two letters**, there should be a delay of **750 milliseconds** while between **two words** there should be a delay of **1000 milliseconds**. For example, suppose we a have string: “AB CD”. The delays between A and B characters and, C and D characters should 750 milliseconds whereas the delay between B and C should be 1000 milliseconds. For terminating the string, you can consider a dot (“.”) as the last symbol.

Part 3

The next step is to write a program for the sender to write the generated Morse code to a pin. If you are using pin 13, then the on-board LED will be blinking.

Part 4

The next step is to write a program to receive the generated Morse code using another Arduino board. The received signal from the pin should be translated into a string and then be printed out to the Serial Monitor of the PC connected to the receiver. Write the receiver code based on the same Morse code protocol (see the Figure 1). You must build the translator by measuring the duration of "0"s and "1"s. Dots and dashes can be detected by measuring duration a "1" and, dot-dash space and individual characters space can be detected by measuring the duration of a "0". So you will implement a polling-based method in first step and an interrupt-based method in second step. You can use the ***millis()*** function to measure the length of a ***HIGH*** or ***LOW*** signal. When you detect a signal change (low to high or high to low by interrupt or polling), you can use a variable to store the first timestamp of when the change is occurred. When you detect the next change, you can use the millis() function again to get the second timestamp and using simple math, you can compute the duration of a "0" or "1". For the interrupt-based implementation, whenever a rising or falling edge is sensed, an interrupt is generated according to the mode you configure and you can use this interrupt to measure the width of a HIGH or LOW signal. This code will be uploaded to the Arduino Nano (the receiver).

This project should be done in two manners:

1. Polling-based:

A wire connects a pin of the ***sender*** to one of digital pins of the ***Receiver,*** where you will write the code to continuously monitor a specific GPIO pin, translate the received Morse code, and print out the received text on the serial monitor. The sender will loop continuously to send the same message again and again and the receiver will loop continuously to monitor and print to the received string unless a dot (".") is received.

2. Interrupt-based:

A wire connects a pin of the ***sender*** to one of the digital pins of the ***Receiver,*** where the receiver detects a rising or falling edge on the pin (by setting the interrupt). You should save the timestamp of the rising/falling edges using millis() function inside the ISR function. In this programming method you shouldn't monitor the pin continuously in the loop function. In order to use interrupt, you please refer to lecture on the Canvas or use this page:

https://www.arduino.cc/en/Reference/AttachInterrupt

Using ***attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)*** function, you can set an interrupt for a specific pin and specify the mode and the ISR function to be called.

- digitalPinToInterrupt(pin) converts the board pin number to interrupt pin number.
- ISR is the name of the interrupt service routine that will be called when an interrupt is occurred.
- Mode specifies the type of interrupt. Four predefined values exist for mode:
    - **LOW**: to trigger the interrupt whenever the pin is low,
    - **CHANGE:** to trigger the interrupt whenever the pin value changes
    - **RISING:** to trigger when the pin goes from low to high,
    - **FALLING:** for when the pin goes from high to low.

  Since you should be able to detect the signal width for both high and low levels, you can set the mode to CHANGE and use a simple check to see if it's a rising or falling edge.

A "dot" is **250 milliseconds** long, and a dash is **750 milliseconds** long. The space between 2 dots or dashes is **750 milliseconds**. In order to make sure the program is able detect dots, dashes and spaces correctly, use a wider range for checking. For example, to detect a dot, write the ***IF*** structure as [150 to 350] (e.g. IF (duration >= 150 && duration <=350)).

The input is a sentence, which is set in the sender program (Arduino Mega 2560), and the output will be showed in the serial monitor window connected to receiver (Arduino Nano) via USB cable.

Submission files:

1. MorseGenerator.ino (on Arduino Mega)
2. MorseGenerator.cpp (on Arduino Mega)
3. MorseGenerator.h (on Arduino Mega)
4. MorseReaderPolling.ino (on Arduino nano to read morse code by polling)
5. MorseReaderInterrupt.ino (on Arduino nano to read morse code by interrupt)
6. MorseReader.cpp (on Arduino nano)
7. MorseReader.h (on Arduino nano)
8. Integrity.txt

You have to convert the files above to the text format (copy and paste their context into the following files in order) and then submit in Canvas system:

1. MorseGenerator.txt
2. MorseGenerator_cpp.txt
3. MorseGenerator_h.txt
4. MorseReaderPolling.txt
5. MorseReaderInterrupt.txt
6. MorseReader_cpp.txt
7. MorseReader_h.txt
8. Integrity.txt

All files will be checked with Canvas plagiarism system.

Since this project has two parts (polling and interrupt), on the demo day, you will be asked to upload the polling-based program and show the result and then upload the interrupt-based program and show the results again.

Attention! Please submit on time. Late submissions will be awarded 0 points.

How the project is graded:

On the demo day, groups are called to present the demo by uploading and running the program. Then, the TA will change the values of **morsePin** on Arduino Mega and/or Nano connected to the PC. Then, you will upload your programs on them respectively when TA asks you to do so. When the program is running on Mega and nano boards, the TA will ask you to give a string by **Serial.read()** function to the Mega board for the **morseString**. Finally, he will check the transferred sentence on the terminal monitor connected to Arduino Nano. TA will ask you to change the pin number and test your code again. The pin should not be defined hard coded. You can define it by "*#define*" preprocessor keyword or constant so that the TA can change it easily at just once and see the result on the corresponding pin on Mega and/or nano boards. The TA will also ask some questions from all members of the group about how the program is written and works, what will happen if you change some things in the program and regarding your understanding of the program and the Arduino platform related to this application.

***Note that**: Even if you have worked on one part of the program, you need know about all parts of the program and be able to answer the questions.*

Grading rubric:

The grading rubric is as follows:

Points	Description
1	The code uses a ***polling*** method to watch the incoming pin.
1	The code uses an ***interrupt*** method to watch the incoming pin.
1	The code properly uses the ***millis()*** function to measure the length of a High or Low signal.
1	The code is well documented and is easy to read.
2	The Morse code reader successfully reads and prints the proper string on the serial monitor using ***polling-based*** technique.
2	The Morse code reader successfully reads and prints the proper string on the serial monitor using ***interrupt-based*** technique.
1	The Morse code reader is able use individual pins (i.e. the pin is not hard coded).

1	One of the group members is able to answer the questions of TA regarding the written program.
-10	If you do not upload your code up to the deadline, you will be graded 0.