**Simultaneous Morse Code Telegraphy using Radio Waves and Arduino UNOs**

Vivswan Shah and Luigi Erba  
Department of Physics, Illinois College, Jacksonville, Illinois 62650

**Abstract**

Morse code was a primitive but revolutionary way used by the military in the late nineteenth century to exchange information. This communications language was originally a wired one-way signaling system between two devices, that is, if one device transmits then the other one can only receive; the two devices cannot transmit or receive at the same time. In this paper Arduino UNOs are employed to overcome this problem and add features like radio transmission, verification of delivery, ability to program different Morse code styles, integrated LCD display, real-time Morse code-to-letter translation and backspace function. The program created by our team and installed on the UNOs can convert Morse code into radio waves and backwards to provide a seamless communication experience.

**1. Introduction**

Since the Morse Code was invented in the 1830s, many ways to transmit it have been proposed and developed. A good communication system must overcome three main obstacles: it must allow simultaneous communication, feature verification of delivery, and have little to no latency to allow a seamless user experience. This paper introduces a wireless telegraphy apparatus made using Arduino UNOs, which solves all the three-problem stated above. Since this apparatus uses radio waves, and radio waves travel at the speed of light, the latency is very low. The simultaneous communication and the verification of delivery can then be achieved using a program running in the Arduino UNO.

A few extra features were added to this concept to improve the user experience even further: an LCD to display and store the sent and received messages in real-time, a backspace functionality to delete and clear the sent or received messages, and a ‘feedback mode’ functionality to switch between acoustic or visual feedback to a message being received.

**2. Background**

The electric telegraph, first developed in the 1830s by Samuel Morse, Leonard Gale and Alfred Vail, was a device used to send messages through a telecommunication circuit using electrical signals [1]. The telegraph was a single-circuit device that worked by pushing the operator key down to complete the electric circuit of the battery. This action sent the electrical signal across a wire to a receiver at the other end. All the system needed was a key, a battery, a wire, a line of track-side poles for the wire and a receiver [1]. In those same years Samuel Morse invented the Morse code, bearing his name, which in conjunction with the electric telegraph revolutionized the way long-distance communication worked. The code featured a set of dots (short mark) and dashes (long mark) for each letter of the English alphabet that allowed complex messages to be transmitted through long distances. The letter codes were assigned based on the frequency of use in common conversations, so that letters such as E, I, S, O, were given short and simples codes whereas letters such as Q, J, Y were left with the longer codes [1].

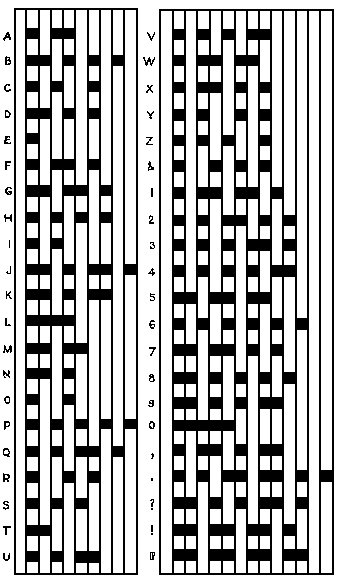


Figure 1. American (Morse) Code

The telegraph system quickly spread across the U.S. and the world, aided by further innovations. Among these improvements was the invention of good insulation for telegraph wires, by Ezra Cornell, and the Quadruplex system, by Thomas Edison, which allowed four messages to be transmitted simultaneously using the same wire [1].

During the 1890s the Morse Code began to be used in conjunction with radio communication, before voice transmission had been invented. When used over radio, its dots and dashes were represented by a series of short and long tones. The invention of the wireless telegraph by Italian inventor and electrical engineer Guglielmo Marconi allowed the Morse Code to become a vital tool for shipping and maritime navigation in the early twentieth century, particularly in emergencies [2]. The code “SOS” was invented because its simplicity in the Morse Code, consisting of three dots, three dashes and three dots again (\*\*\* --- \*\*\*).

**3. Main Device**

**3.1 Components**

Each of the two radio communication devices was made using the following components:

1. Arduino UNO
2. Breadboard
3. Breadboard-compatible components:
   1. 1x 16x2 LCD Display Module (LCM1602C)
   2. 1x HC-12 Wireless Serial Port Module 1000m (HC-12)
   3. 1x HC-12 Compatible Antenna
   4. 1x Buzzer (PKM22EPP-40)
   5. 1x LED Light
   6. 3x Tactile Switches (K47)
   7. 2x 220-ohm Resistors
   8. 3x 10k-ohm resistors
   9. 43x Male/Male Jumper Wires

**3.2 Use of the components**

1. **Arduino UNO:**

An Arduino UNO can be thought of as a processor with a flash memory of 32 KB . This component provides the processing capacity and power that is needed to manipulate and covert the user input (the Morse Code input) to the desired output (radio waves). It is also able to convert the Morse Code into alphabetical letters or numbers and add features specified by the source code our team designed, such as verification of delivery, backspace functionality, simultaneous sending and receiving troubleshooting and optimization. The processor and source code also allow for different modalities of ‘transmission received’ feedback, allowing you to choose between acoustic feedback only (buzzer), visual feedback only (LED light), or audio and visual feedback together (buzzer and LED light).

1. **HC-12 Wireless Serial Port Module:**

The HC-12 module has five pins: TX, RX, SET, GND, VCC. GND and VCC are used to provide power to the HC-12. TX is a transmission pin, it takes the data from the Arduino UNO in the form of pulsating digital electric waves and converts it into radio waves. The radio signals received by the HC-12 are converted to pulsating digital electric waves, which are then sent to the Arduino UNO through the RX pin, also known as receiving pin.

1. **LCD Display Module:**

The use of an LCD display is for both debugging purposes and to provide a visual interpretation of the Morse Code messages sent and received. The LCD Display also gives some clues to the person running the device regarding the status of the device and transmission; if the device is running properly the LCD Display will be always ON and will feature an “S:” row, documenting the message ‘SENT’ the device, and an “R:” row, documenting the message ‘RECEIVED’ by the device. If at any time the “S:” is replaced by an “X:”, it means that there was a problem with the transmission and that the message was not sent successfully. The LCD Display Module can display as much as 16 consecutive letters received or sent before scaling the windows forth as the message lengthens.

1. **Buzzer:**The buzzer is used to give an acoustic feedback of a message being received by the device.   
   It also allows for the person running the device to know that the device is in fact working properly and transmitting. The buzzer can be used for acoustic feedback or can be turned off by switching to the “LED light only” mode.
2. **LED Light:**

The LED light is used give a visual feedback of a message being received by the device. The light will flash during a transmission meaning that a message was successfully received. The LED light will not flash in any other occasion. The LED light can be turned off by switching to the “Buzzer only” mode.

* 1. **Assembling and Design**

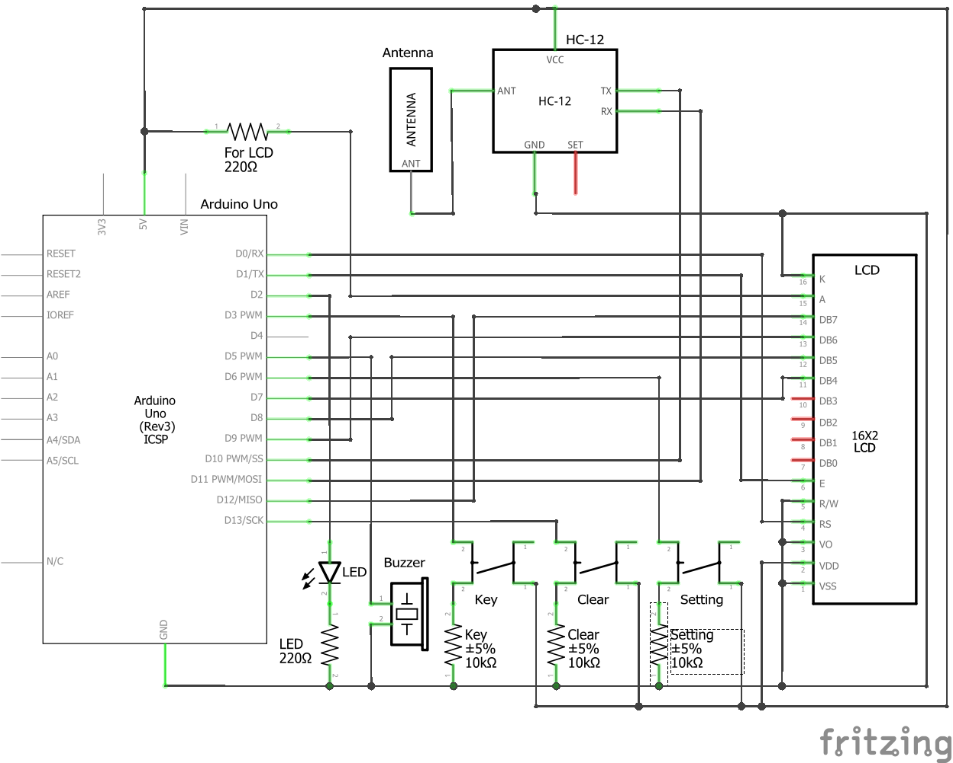
1. **Start by connecting the LCD Display Module to the Arduino UNO:**
   1. Connect **VSS** (**PIN 1**) to **Ground**, since VSS is the ground or negative supply for the LCD processor.
   2. Connect **VO** (**PIN** **3**) to **Ground**, for full LCD screen contrast.
   3. Connect **R/W** (**PIN** **5**) to **Ground**, to set the LCD to ‘always writing mode’.
   4. Connect **K** (**PIN** **16**) to **Ground**, to provide negative (ground) supply to LEDs in LCD.
   5. Connect **A** (**PIN** **15**) to the **220-ohm resistor**, which is connected to the **5V** supply. This is because A is the positive supply for the LED of the LCD display and the 220-ohm resistor decreases the current to the desired level.
   6. Connect **DB4**, **DB5**, **DB6**, **DB7** (**PIN** **11**, **12**, **13**, **14**) of the LCD, also known as data pins, to **D7**, **D8**, **D9**, **D12** of Arduino UNO.
   7. Connect **RS** (**PIN** **4**) to **D0/RX** to enable the recording memory of the LCD display.
   8. Connect **E** (**PIN** **6**) to **D1/TX** to enable the recording of the writing.
2. **Then connect the HC-12 Wireless Module to the Arduino UNO:** 
   1. Connect **GND** (ground supply) to **Ground**.
   2. Connect **VCC** (positive supply) to **5V**.
   3. Connect **ANT** to the HC-12 compatible **antenna**.
   4. Connect **TX** to **D10** **PWM**. This is because a pulsating digital signal is needed for the transmission, and all pulsating digital connections are connected to PWM.
   5. Connect **RX** to **D11** **PWM**. For the same reason as above, the receiving signal from the HC-12 will be a pulsating digital, and all the pulsating digital connections are connected to the PWM PINs.
3. **Connect the LED to the Arduino UNO:**
   1. Connect **positive** terminal of LED to **D2**.
   2. Connect **negative** terminal of LED to **220-ohm resistor**. This is to decrease the current through the LED and prevent it from burning.
   3. Connect the **220-ohm resistor** to **Ground**.
4. **Connect the Buzzer to the Arduino UNO by connecting one terminal to the D5 pin (to provide a pulsating digital signal) and the other terminal to Ground.**
5. **Connect the three Tactile Switches (Key, Backspace, Feedback Mode):**
   1. Connect **PIN 2** of Key, Backspace, Feedback Mode to **D3**, **D13**, **D6** respectively.
   2. Connect **PIN** **3** of Key, Backspace, Feedback Mode to **5V**.
   3. Connect **PIN** **1** of Key, Backspace, Feedback Mode to **10k-ohm resistor** so that approximately no current passes through them. This is necessary to reduce the electrical noise in the switches by providing a base line.

Figure 2. Schematic concept of each device

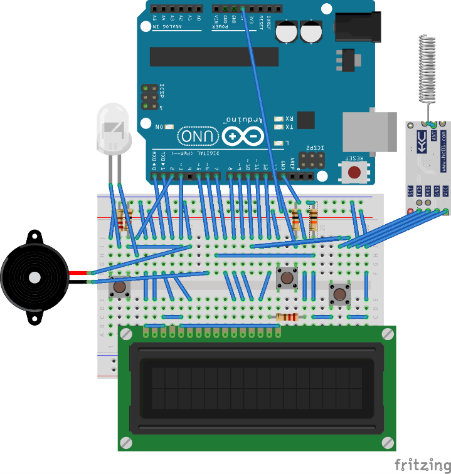


Figure 3. Two-dimensional design of each device

* 1. **Program design**

The programmatical constants used were the following:

#define DOT\_INTERVAL 150

#define BUZZER\_FREQ 700

#define MORSE\_READ\_RATE 2

*Code 1. Changeable constants of the program.*

1. **DOT\_INTERVAL:** This constant’s unit of time is milliseconds.If the key is pressed for a period of time shorter than 150 milliseconds, the press will be registered as a dot, else as a dash.
2. **BUZZER\_FREQ:** This constant represents the frequency of the sound emitted by the buzzer in Hertz.
3. **MORSE\_READ\_RATE:** This constant regulates the amount of time the acoustic or visual feedback will be exhibited for. The amount of time for which the buzzer will sound and the LED will shine when a message is received is the result of a proportion between this constant and the DOT\_INTERVAL, DASH\_INTERVAL, or WORD\_INTERVAL.

#define DASH\_INTERVAL DOT\_INTERVAL \* 3

#define WORD\_INTERVAL DOT\_INTERVAL \* 7

*Code 2. Dependent constants of the program.*

1. **DASH\_INTERVAL:** This constant is set to as three times of the DOT\_INTERVAL. It represents the amount of time of acoustic and/or visual feedback in response to a dash code. For our experiment it was set to be 450 milliseconds.
2. **WORD\_INTERVAL:** This constant is set to as seven times of the DOT\_INTERVAL. It represents the amount of time the device will wait before registering the new incoming code as a new word, as opposed to the same word (no spaces in between the codes). For our experiment it was set to be 1,050 milliseconds.

void loop() {

setting();

loopEditButton();

loopSettingButton();

loopMorseKey();

loopLCD();

loopBuzzer();

loopHC12();

}

*Code 3. Loop method of the program.*

The main loop of the program contains the following functions:

1. **setting:** This function allows the device to delete the data that is not important or necessary in order to save memory. This is crucial since Arduino UNO is a very low memory device.
2. **loopEditButton:** This function checks and records how many times the clear button is pressed in a pre-established time period. Pressing the clear button multiple times and in the same or different time periods will results in different actions.
   1. **1 Time**:Acts as a backspace key and removes the previously typed letter.
   2. **2 Times:** Clears the screen from all the sent messages.
   3. **3 Times:** Clears the screen from all the received messages.
3. **loopSettingButton:** This function changes the mode between: both light and sound, only sound and only light, for every press of the setting button by user in order.
4. **loopMorseKey:** This function converts the user input through the key button (Morse Code) into text that can be read by the HC-12 and translated by it into radio waves. The device has been set to write and read the International Morse Code (ITU) variant.

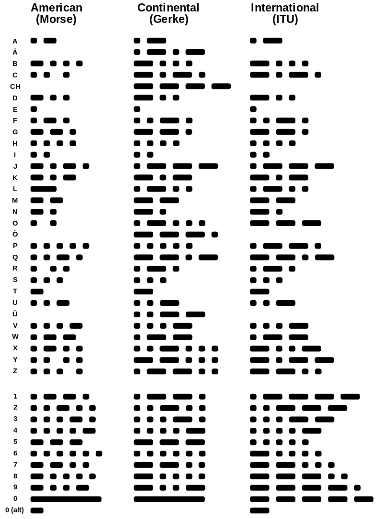


Figure 4. Morse Code variants.

1. **loopLCD:** This function manages the LCD display and updates it every processing cycle, to allow the messages received and sent to be updated and shown instantly on the LCD.
2. **loopBuzzer:** This function decides whether the buzzer needs to beep or not and sends the signal to the buzzer accordingly.
3. **loopHC12:** This function checks whether any new data is received by the HC-12 module. If any new data is received, it converts it into text and saves it so that the LCD can be updated in the next cycle and show the message received.

**4. Observations**

After having assembled the device, during the testing phase, we observed few phenomena.

First observation: in an open field, the device worked continuously until around 100 m of distance between the devices. After around 100 m, the devices occasionally failed to send or receive messages. However, if the antennae of the devices were pointed toward each other the range increased to around 150 meters. This tells us that the antenna used in conjunction with the HC-12 is a directional antenna.

Second observation: during the programming phase of the devices it was observed that if both the devices were transmitting simultaneously none of them would receive any data, which suggested that the HC-12 module can only act as a receiver or a transmitter at a given time, not both. This problem was overcome by introducing a function in the source code that rapidly switches between the receiving and the transmitting mode, therefore lowering the chances of failure of transmission.

Third observation: after 150 m of distance between the devices, as we increase the distance even more, the ratio between successful transmission and unsuccessful transmission decreases very rapidly, suggesting that the intensity of the radio waves perceived by each device was decreasing. Since the intensity of radio waves from a point source is inversely proportional to the square of the distance between them, and since the size of the antenna is very small compared to the distance between the devices, we can deduce that the intensity of the signal is decreasing very rapidly due to this phenomenon.

**5. Conclusion**

The creation of these functioning devices demonstrates that a simultaneous wireless telegraphic communication system is possible even when the antennae (HC-12) used can only transmit or receive at a given time. This is possible by adding a function to the source code that rapidly switches between the two states (transmitting and receiving). This project also challenged us with the creation of a communication system that does not rely on the internet, on which many communication devices are based nowadays.

Even though the devices work properly and reliably, this project is still a prototype and may contain some unforeseen bugs in the source code; there could also be some improvements to the user experience and to the structural integrity of the device.

**6. Acknowledgements**

Our acknowledgements go to Illinois College’s Department of Physics for the providing us with the funds and giving us access to the physics laboratory, to Dr. Pratheesh Jakkala for giving us the idea for this project and supporting us during its development, and to Arduino for its amazing products and resources.

**7. Resources**

The schematics, source codes and instructions related to the prototypic devices have been uploaded to the GitHub community in order for this project to be developed further and improved by the public in an unrestricted, open-source, and proprietary-safe environment.

Link to GitHub Page: [*https://github.com/Vivswan/MorseCodeRS*](https://github.com/Vivswan/MorseCodeRS)

# **References**

**1.** Editors, H. (2018, August 21). *Morse Code & the Telegraph*. (A&E Television Networks) Retrieved December 11, 2018, from HISTORY: https://www.history.com/topics/inventions/telegraph

**2.** Editors, T. (2017, May 4). *A landmark demonstration of Morse Code telegraphy*. Retrieved December 11, 2018, from The Telegraph: https://www.telegraph.co.uk/technology/connecting-britain/significant-demonstration-morse-code/

**3.** Arduino. (2018, October 28). *Documentation*. Retrieved December 11, 2018, from Arduino.cc: https://www.arduino.cc/

**Fig. 1** White, T. H. (2005). *Cyclopedia of Applied Electricity* (1911 ed., Vol. 6). American School of Correspondence. Retrieved December 11, 2018

**Fig. 2** Huurdeman, A. A. (2003). *The Worldwide History of Telecommunications* (1 ed., Vol. 1). Wiley-IEEEE Press. Retrieved December 11, 2018