LDFA's ESA project: morse code device

A more serious feasibility study

University: Unicam, Università degli Studi di Camerino
Institute: Scuola di Scienze e Tecnica
Course: Master in Computer Science
Teaching: Embedded Systems Architecture
Professors: L. Morresi, A. K. Piermartei
Student: Luciano De Falco Alfano
Project: LDFA's ESA project: morse code device

Document public draw version 0.1 on 29th jan 2019

General index

Scope

[What problem am I trying to solve? What is the aim? What is the background?]

Goal

Our goal is design and implement a *Morse Code Device*. That is a micro controlled system able to output to a buzzer a string of characters sent from the user. Buzzer will play the characters using morse code. Moreover it must be able to sound directly code operated by user using a button.

Hereafter our device is named as MCD (Morse Code Device).

User sends characters to play to MCD using a Bluetooth connection between a mobile phone and the device.

Background

Morse Code is a method to encode text using two signals of different time duration [see ref. 3]. These signals are called *dots* (with short duration) and *dashes* (with long duration).

Given a unit of duration called (time) unit:

- *dots* last 1 unit,
- *dashes* last three units [see ref. 1];
- elements (dots and dashes) composing a character are separated by 1 unit of silence; this is called *intra-character* space;
- between two characters there is an *inter-character* space, this lasts 3 units;
- finally, between two words there is a *word* space that lasts 7 units.

To learn the morse code memorizing the numbers of dots and dashes making every single character, and try to count them hearing at the arriving signal is inefficient.

Morse code operators store and recognize directly the “rhythm” of every character, without disassembling it by its elements.

The first method (counting dots and dashes) is effective only up to a rate of 13 words/minute (wpm). While the second method (recognize directly the pattern of sound of a character) is efficient far more than 20 wpm.

To memorize a sound pattern of a character, a morse operator refers to ref. 2 where every single character is showed using *dits* (pronounced *di* or *dit*), and *dahs* (pronounced *dah*, using an extended *a*), permitting an onomatopoeic pronunciation of the coded characters. Memorization happens grabbing the sequence of tones of the elements, and not through their duration.

Speed of transmission is measured using the word (with an appending end space) “PARIS “. This translates to "`.--. .- .-. .. ...`/". Counting units for dits, dahs, intra and inter character spaces, and the final word space, we get a total of 50 units. Transmitting using a (standard) rate of 20 wpm means to be able to transmit “PARIS “ 20 times every minute.

If s_{wpm} is a wanted transmission rate, then our *unit* needs to be: $t_{dit} = 60/(50 \cdot s_{wpm})$ seconds. For example, if $s_{wpm} = 20$ then $t_{dit} = 0.06$ s.

System design

[What do I want to make? What are the features? How should it be used?]

As a first stage, we design a system composed of an actuator device, capable “to play” morse code using a buzzer. This is the MCD.

This device needs to be able:

1. to set a wanted *wpm* to use in subsequent transmissions;
2. to receive a string of characters to play;
3. to play the received string;
4. during the playing, show what character is played;
5. permit to a user to play directly morse code using a button.

From the previous functionalities, we need a mean to communicate to MCD. To this purpose we use a Bluetooth device. It connects MCD to a mobile phone. This phone will host a simple app to send commands to MCD. Available commands are:

a) set *wpm* to use from now on;

b) play a string written by user;

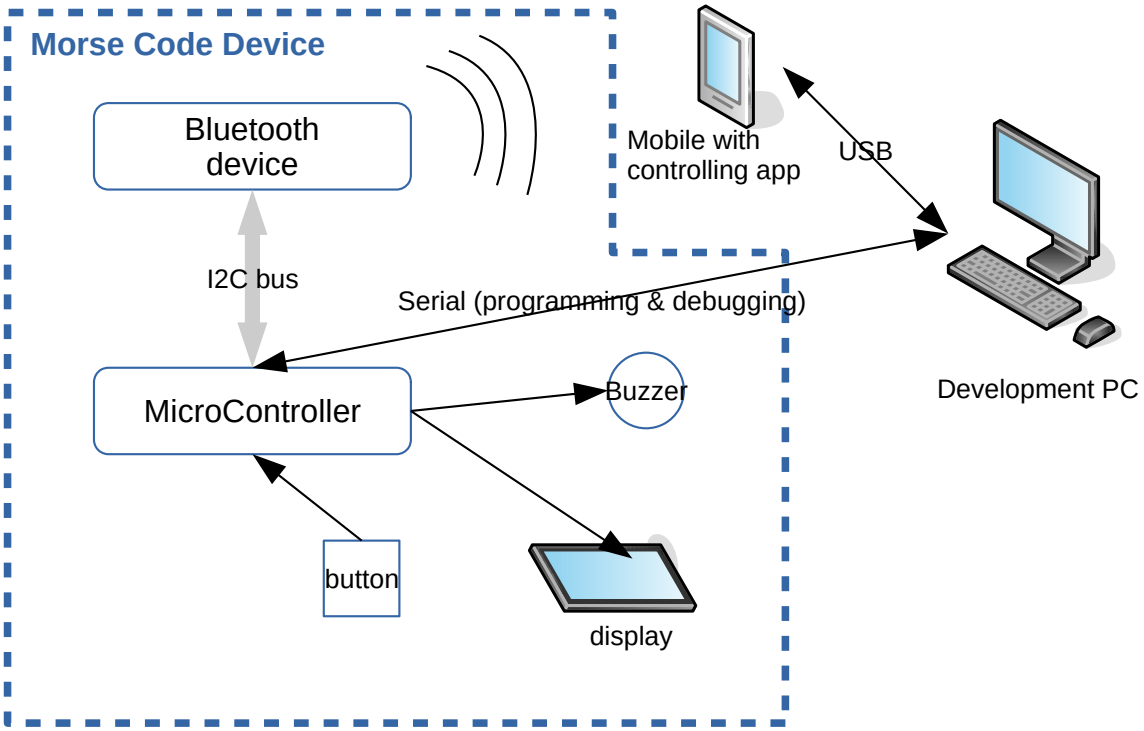
c) play character by character written by user;

d) play *dit*/*dah*.

Implementation

[what devices to use? how do they communicate?]

The following diagram shows the general architecture of the project.



Picture 1: Project architecture

So, we can split our project into these environments:

- the MCD device; it controls the physical components to play morse code and to interact with external environments (mobile and development workstation);
- the mobile; it sends or receive commands from MCD, and if necessary it plays morse code signals from MCD input device (the operator's button);
- the development PC; it hosts the development environments to implement the necessary code for Arduino in MCD and for the mobile phone.

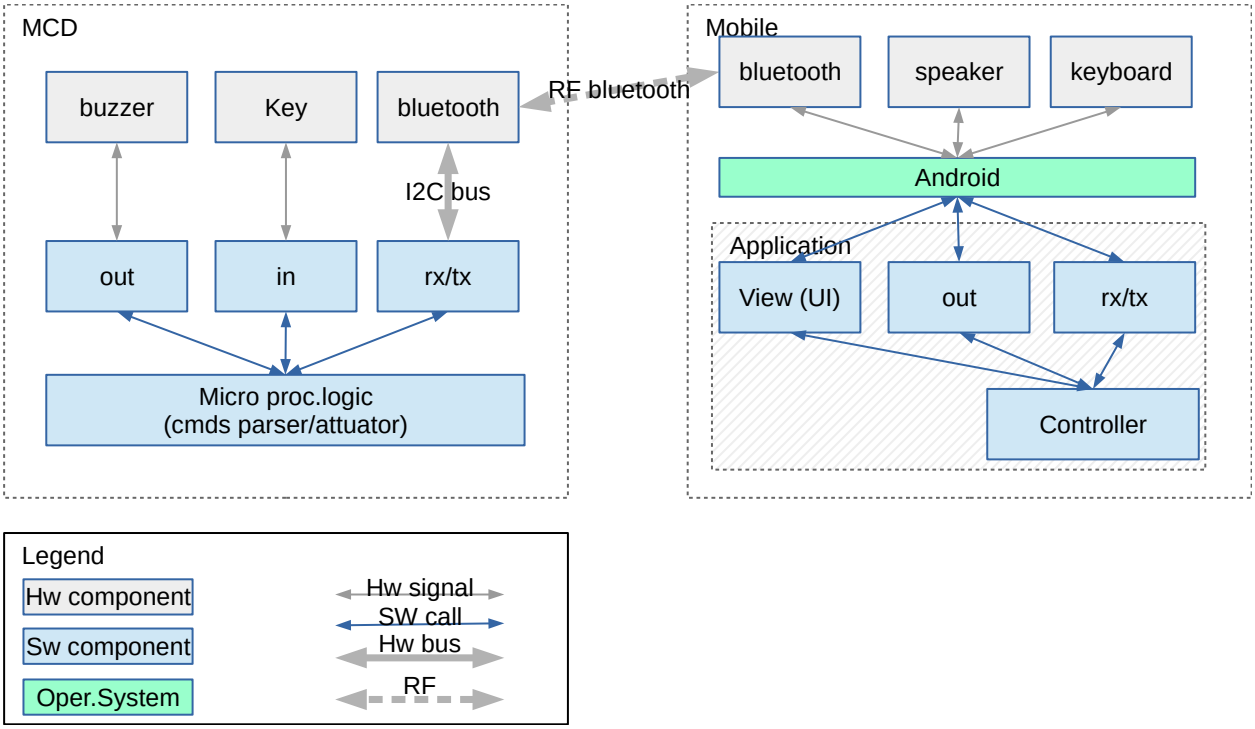
Constraints

Main constrains about our projects are:

- MCD micro controller is an *Arduino UNO R3* or an *Arduino Mega 2560* (originals or compatible);
- Development workstation operating system is *Windows 10*;
- Mobile phone operating system is *Android*.

Partitioning systems in subsystems

We can decompose these environments in (high level) subsystems as shown in the following image.



Picture 2: Subsystems

Hereafter we define them better.

MCD device subsystems

Micro processor logic is the high level logic controlling other MCD modules. This module receives commands from *mobile* through *rt/tx* module, interprets them and actuate them using *in* and *out* modules. In case, it sends commands to *mobile*, using *rx/tx* module.

Out module plays morse code, converting a phrase from ascii characters to acoustic signals. Moreover, it administrate global parameters, for example the duration of the (time) *unit*.

In module, analyzes and standardizes a signal from a key manually operated, converting it in a series of dots, dashes, and spaces (intra/inter-character, word spaces). It can convert these information in ascii codes, if requested.

Rx/tx module drives the bluetooth hardware module to communicate with the linked mobile phone.

Mobile phone subsystems

To design.

Development PC subsystems

About the development workstation, we can organize it as:

- *Arduino IDE*;
- *Android Studio*.

Subsystem: MCD logic

To design.

Subsystem: MCD out

To design.

Subsystem: MCD in

To design.

Subsystem: MCD rx/tx

To design.

Subsystem: Mobile controller

To design.

Subsystem: Mobile view

To design.

Subsystem: Mobile out

To design.

Subsystem: Mobile rx/tx

To design.

Acronyms, abbreviations and references

Acronyms, abbreviations

Acronym	Note
CW	Continuous Wave
IDE	Interactive Development Environment
MCD	Morse Code Device
UI	User interface
wpm	Words per Minute

References

Ref.	Title	Note
1	Morse Code Timing	The bases of timing the code, at https://morsecode.world/international/timing.html
2	International Morse Code	A code reference, at https://morsecode.world/international/morse.html
3	International Morse Code (2)	Another code reference using dashes and dots, at https://morsecode.world/international/morse2.html
4	Learning Morse Code as a Language	From J.A.Ritter WØUCE, ver.2.6 June 2015, at https://www.kb6nu.com/wp-content/uploads/2017/11/LearningMorseCodeasaLanguageVersion2.6June2015.pdf

Ref.	Title	Note
5	Morse Code	at https://en.wikipedia.org/wiki/Morse_code
6	Android studio download	Android SDK, download site at https://developer.android.com/studio/#downloads

Functions checklist

Id	description	Implemented in
1	set a wanted wpm to use in subsequent transmissions;	
2	receive a string of characters to play;	
3	play the received string;	
4	during the playing, show what character is played;	
5	permit to a user to play directly morse code using a button.	