CODICE MORSE

1 Introduzione

1.1 Informazioni sul progetto

Titolo del progetto:	Codice morse
Alunno:	Alessandro Gomes, Federico Agosta, Patrick Sartori, Davide Paradiso
Classe:	I3BB/I3AA
Data d'inizio:	10.11.2017
Data di consegna:	19.01.2018
Docente responsabile:	Massimo Sartori

1.2 Abstract

Our school needs a project to exhibit to Espoprofessioni so they commissioned to us the realization of a web page that translates a message to Morse code,
the coded string is sent to an Arduino that will pour water in a tube, a water drop will mean a dot and a longer stream of water will mean a line, the water
will be read by a second Arduino and the message formed by the water will be translated back to the original message, the final result will be displayed on an LCD display.
Our solution will use an html page that has an input tag that collects the original message, the message will be coded using JavaScript, we will be using php to write the message in a database, we will be using two Arduino boards, one to send the message with water and one to translate the message back.

1.3 Scopo

Lo scopo principale di questo progetto è quello di prepararci per i progetti di fine scuola, per fare ciò ci viene richiesto di sviluppare un applicativo in cui si possa comunicare tramite codice morse. Per riuscire a sviluppare questo progetto dobbiamo utilizzare quanto appreso in questi tre anni di formazione. Lo scopo secondario di questo progetto è quello di poterlo mostrare ad Espoprofessioni di quest'anno.
Ci viene richiesto di creare un sistema di comunicazione tra una pagina web e una struttura comandata da due Arduini. Nella pagina web si dovrà inserire del testo che verrà codificata in codice Morse, il testo codificato verrà salvato in un database e, allo stesso tempo, mandato la codifica alla struttura. Il primo Arduino della struttura sarà quello che riceve il testo codificato e, in base a ciò che legge, fa aprire per qualche secondo l'elettrovalvola in modo tale da fare uscire la quantità d'acqua giusta (un punto equivale ad una goccia, una riga equivale ad una "corrente" d'acqua). Un sensore si occuperà di leggere le gocce e di mandare i dati ad un secondo Arduino che si occuperà di decodificare il testo codificato e di stamparlo su un display.

1.4 Analisi

1.4.1 Analisi del dominio

In rete esiste già qualcuno che ha fatto il nostro progetto però con una differenza e cioè che nel progetto già esistente usavano una tastiera direttamente collegata al primo Arduino, mentre a noi viene richiesto la creazione di una pagina web da cui verranno trasmesse le frasi/parole codificate all'Arduino e anche ad una banca dati esterna.
Riferimento: https://hackaday.io/project/10884-arduino-networking-using-morse-code-over-water

1.4.2 Analisi e specifica dei requisiti

ID: REQ-001	
Nome	Realizzo un applicativo che possa comunicare in codice Morse
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Deve essere presente una pagina web dove inserire la frase da trasmettere
002	Deve essere presente una struttura di supporto all'impianto idroelettrico

ID: REQ-002	
Nome	Realizzare un sito che permetta la codifica e la trasmissione del codice Morse
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il sito deve presentare una pagina con l'apposito campo per l'inserimento del testo
002	Il sito deve presentare una pagina con l'apposito bottone per la trasmissione e la codifica del teso
003	Il sito deve presentare una pagina con l'apposito campo per il testo codificato in Morse

ID: REQ-003	
Nome	Realizzazione una seconda pagina web che rappresenterà lo storico del sito
Priorità	2
Versione	1.0
Note	
Sotto requisiti	
001	Il sito deve presentare una pagina con un apposito campo che permetterà all'utente di ricercare una parola o una frase a sua scelta.
002	Il sito deve presentare una pagina con al suo centro la lista di tutte le frasi o le parole inserite.

ID: REQ-004	
Nome	Interfaccia grafica del sito
Priorità	2
Versione	1.0
Note	
Sotto requisiti	
001	Il sito deve essere intuibile e facile da usare
002	Il sito dovrà avere una "barra di navigazione" che permetterà all'utente la navigazione tra la pagina traduttore e la pagina storico

ID: REQ-005	
Nome	Materiali richiesti per la realizzazione del progetto
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	1 Arduino One e 1 Arduino Ethernet per gestire tutti i vari componenti del progetto
002	1 elettrovalvola che permette il passaggio dell'acqua
003	1 pompetta che permette il "risucchio" dell'acqua
004	1 tubo trasparente che permette la visualizzazione delle gocce d'acqua
005	1 ricevitore e 1 emettitore che legge le gocce che cadono all'interno del tubo
006	1 display che mostra il codice morse letto dal ricevitore

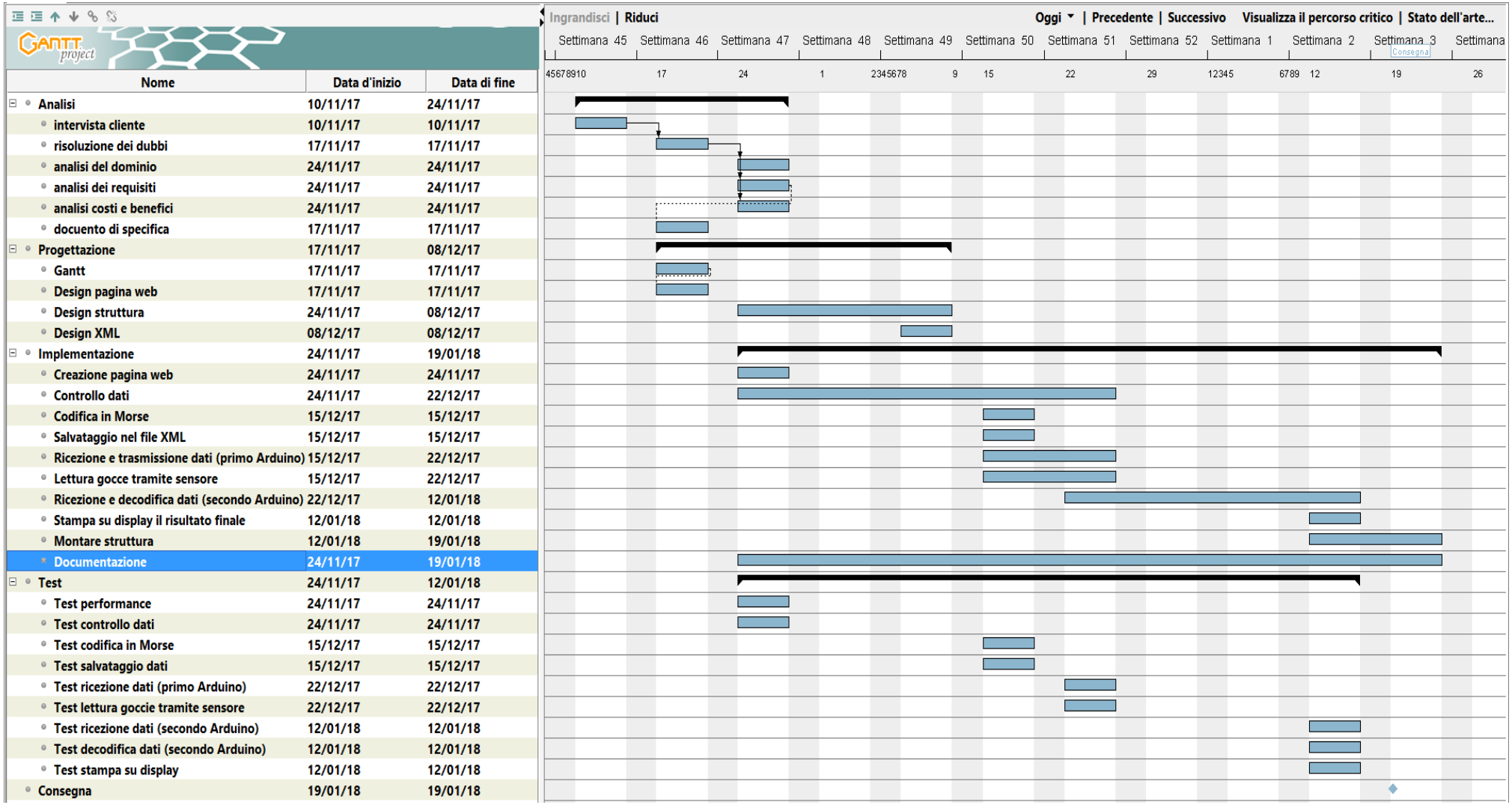
006	Una struttura portante che sosterrà tutti i componenti

ID: REQ-006	
Nome	Database
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Ci deve essere un database esterno
002	Si deve inserire tutte le frasi o le parole all'interno del database

1.4.3 Analisi dei costi

Materiale	Costo
Arduino Ethernet	CHF 50.00
Arduino UNO	CHF 25.00
Piccolo materiale ferramenta (viti, dadi ecc)	CHF 25.50
MTF per struttura e i due pannelli.	CHF 19.50
Stampa 3D flange	CHF 2.50
Tubo plexi diam 120 x 144mm x 1000mm	CHF 25.80
Tubo silicone	CHF 4.80
Dischi trasparenti	CHF 15.60
Coll acrilica	CHF 9.80
Pompa ad induzione da 12 V	CHF 10.00
Alimentatore moderabile USPS-600	CHF 15.00
Adattatori Gardenia + rubinetti	CHF 9.50
Elettrovalvola	CHF 8.40
TOTALE:	CHF 221.40

1.4.4 Pianificazione



Nome	Data d'inizio	Data di fine
Analisi	10/11/17	24/11/17
intervista cliente	10/11/17	10/11/17
risoluzione dei dubbi	17/11/17	17/11/17
analisi del dominio	24/11/17	24/11/17
analisi dei requisiti	24/11/17	24/11/17
analisi costi e benefici	24/11/17	24/11/17
docuento di specifica	17/11/17	17/11/17
Progettazione	17/11/17	08/12/17
Gantt	17/11/17	17/11/17
Design pagina web	17/11/17	17/11/17
Design struttura	24/11/17	08/12/17
Design XML	08/12/17	08/12/17
Implementazione	24/11/17	19/01/18
Creazione pagina web	24/11/17	24/11/17
Controllo dati	24/11/17	22/12/17
Codifica in Morse	15/12/17	15/12/17
Salvataggio nel file XML	15/12/17	15/12/17
Ricezione e trasmissione dati (primo Arduino)	15/12/17	22/12/17
Lettura gocce tramite sensore	15/12/17	22/12/17
Ricezione e decodifica dati (secondo Arduino)	22/12/17	12/01/18
Stampa su display il risultato finale	12/01/18	12/01/18
Montare struttura	12/01/18	19/01/18
Documentazione	24/11/17	19/01/18
Test	24/11/17	12/01/18
Test performance	24/11/17	24/11/17
Test controllo dati	24/11/17	24/11/17
Test codifica in Morse	15/12/17	15/12/17
Test salvataggio dati	15/12/17	15/12/17
Test ricezione dati (primo Arduino)	22/12/17	22/12/17
Test lettura goccie tramite sensore	22/12/17	22/12/17
Test ricezione dati (secondo Arduino)	12/01/18	12/01/18
Test decodifica dati (secondo Arduino)	12/01/18	12/01/18
Test stampa su display	12/01/18	12/01/18
Consegna	19/01/18	19/01/18

1.4.5 Analisi dei mezzi

1.4.5.1 Software

- GanttProject - 2.8.5: utilizzato per fare il gantt preventivo e il consuntivo.
- Bootstrap – 4: utilizzato per realizzare tutto quello che riguarda la grafica del sito (pagina principale e quella dello storico).
- Arduino - 1.8.5: utilizzato per gestire l'elettrovalvola, gestire la lettura delle gocce e infine la decodifica e la stampa del testo non codificato.
- XAMPP - 7.0.26: utilizzato per gestire PHP e SQL.
- SketchUp – 2015: utilizzato per la creazione della struttura completa del progetto completa di struttura di supporto.
- Fritzing - 0.9.3b: utilizzato per la creazione del design dello schema elettrico dei due Arduini.
- Microsoft Office - 2.016: utilizzato per la creazione della documentazione e della presentazione del progetto.
- HeidiSQL - 9.4: Utilizzato per la gestione del database.
- Atom - 1.23.1: utilizzato per la stesura del codice php.
- PhpStorm – 2017.2.4: utilizzato per la stesura del codice php.

1.4.5.2 Hardware

- Arduino Ethernet
- Arduino One
- 2 computer hp
- Computer lenovo
- Computer Asus
- Modelcraft flui pump
- Elettrovalvola

2 Progettazione

2.1 Design dell'architettura del sistema

Questo progetto è composto da un index.php che conterrà la parte grafica iniziale di questo progetto, questo index andrà a richiamare un file esterno. Questo file php andrà ad inviare i dati ricevuti dalla pagina principale ad un database esterno e in contemporanea ad una struttura gestita da due Arduini.

2.2 Design database

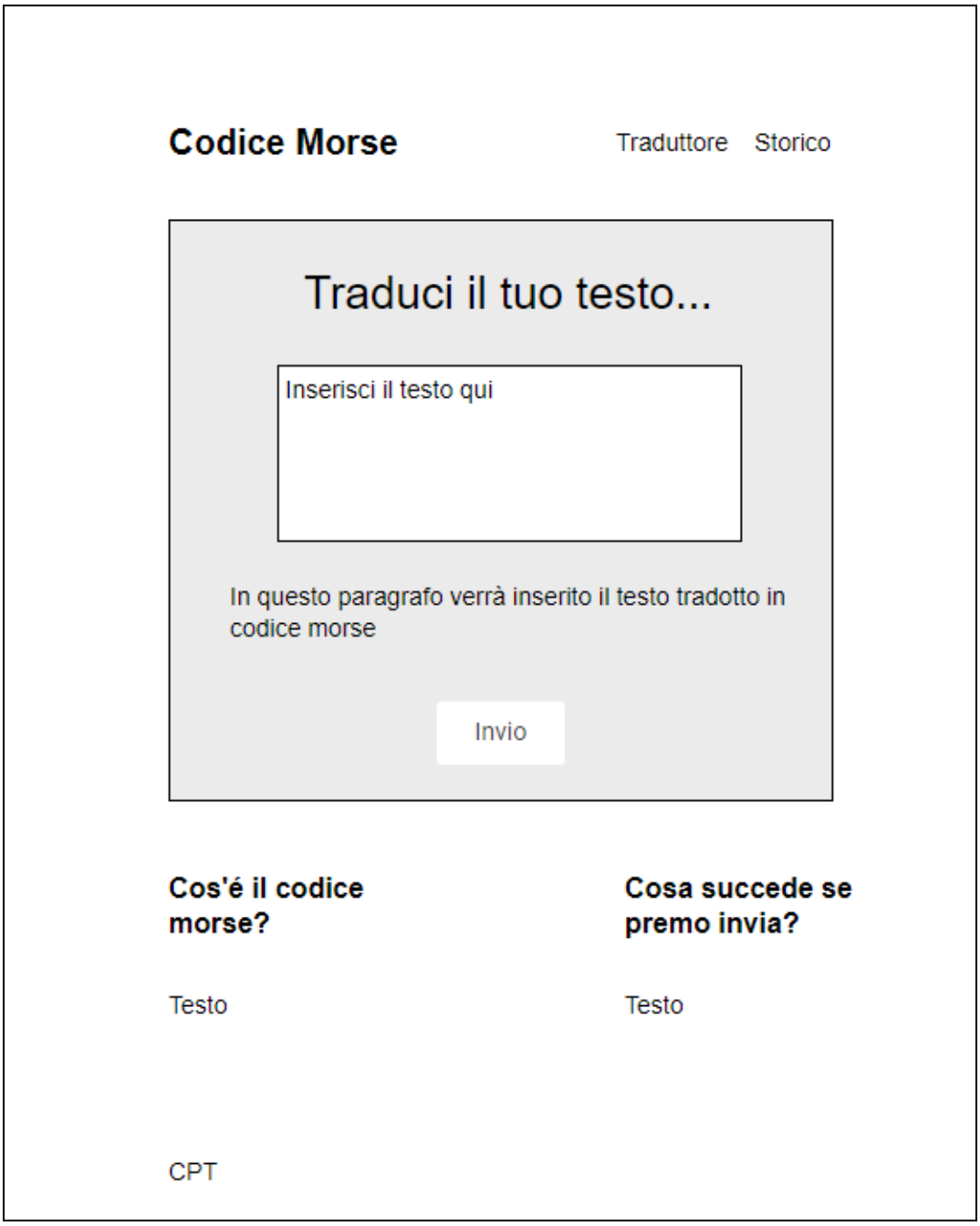
code	
id	int
encoded	text
decoded	varchar(255)

2.2.1 Spiegazione database

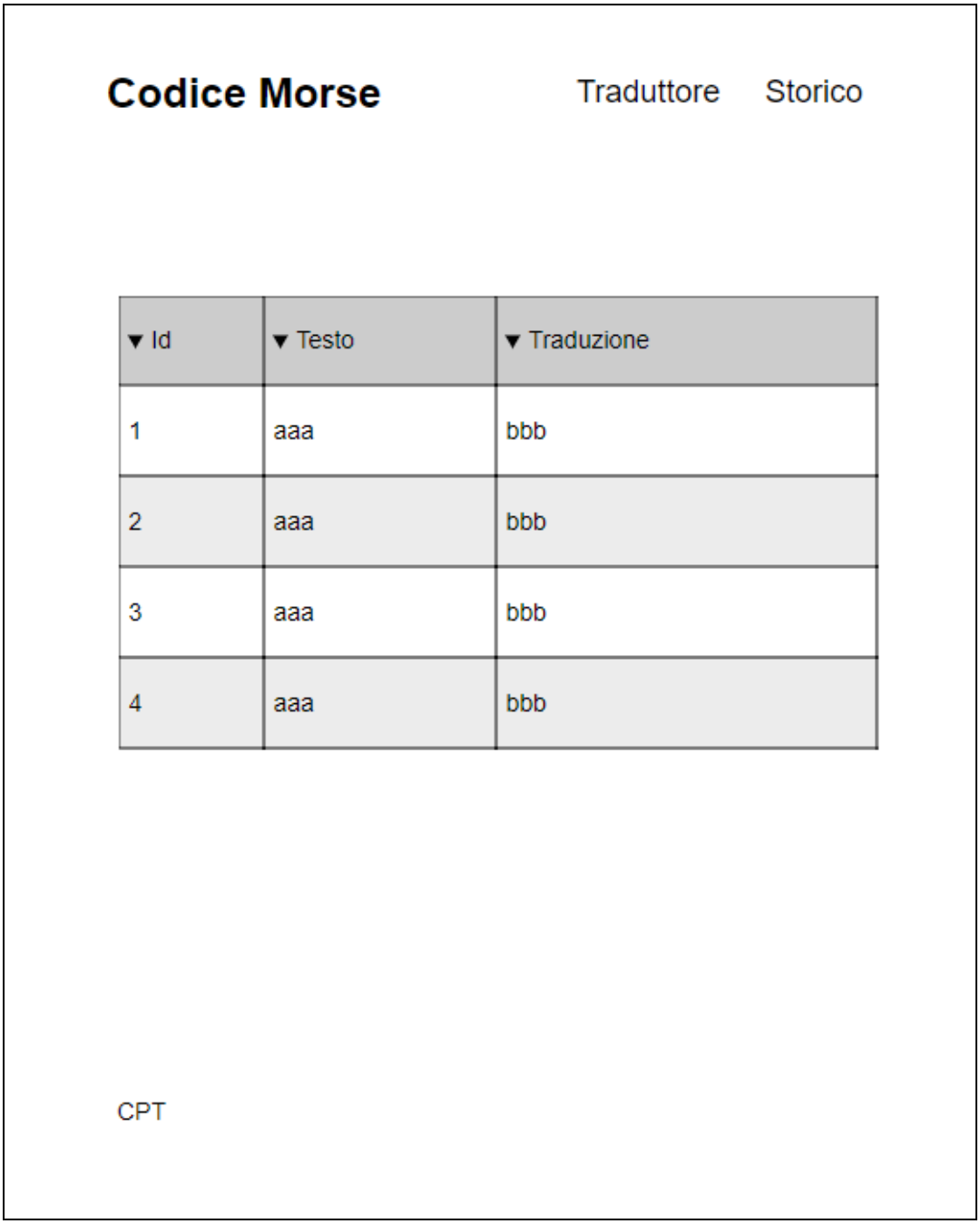
- Id: id della parola o della frase presente nel database.
- Encoder: la parola o la frase codificata in codice morse.
- Decoder: la parola o la frase decodificata.

2.3 Design delle interfacce

2.3.1 Design index



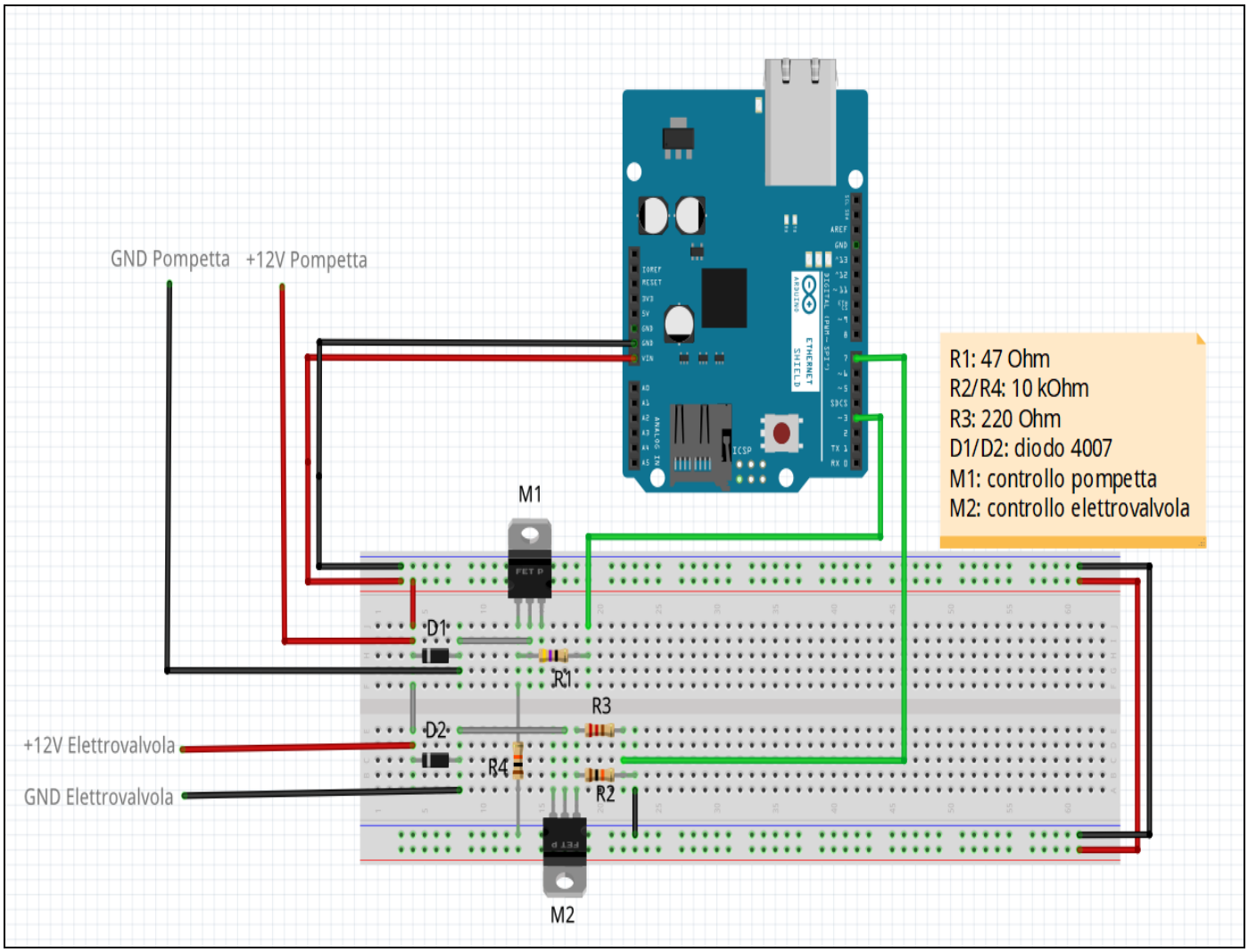
2.3.2 Design storico



2.4 Design schema elettrico

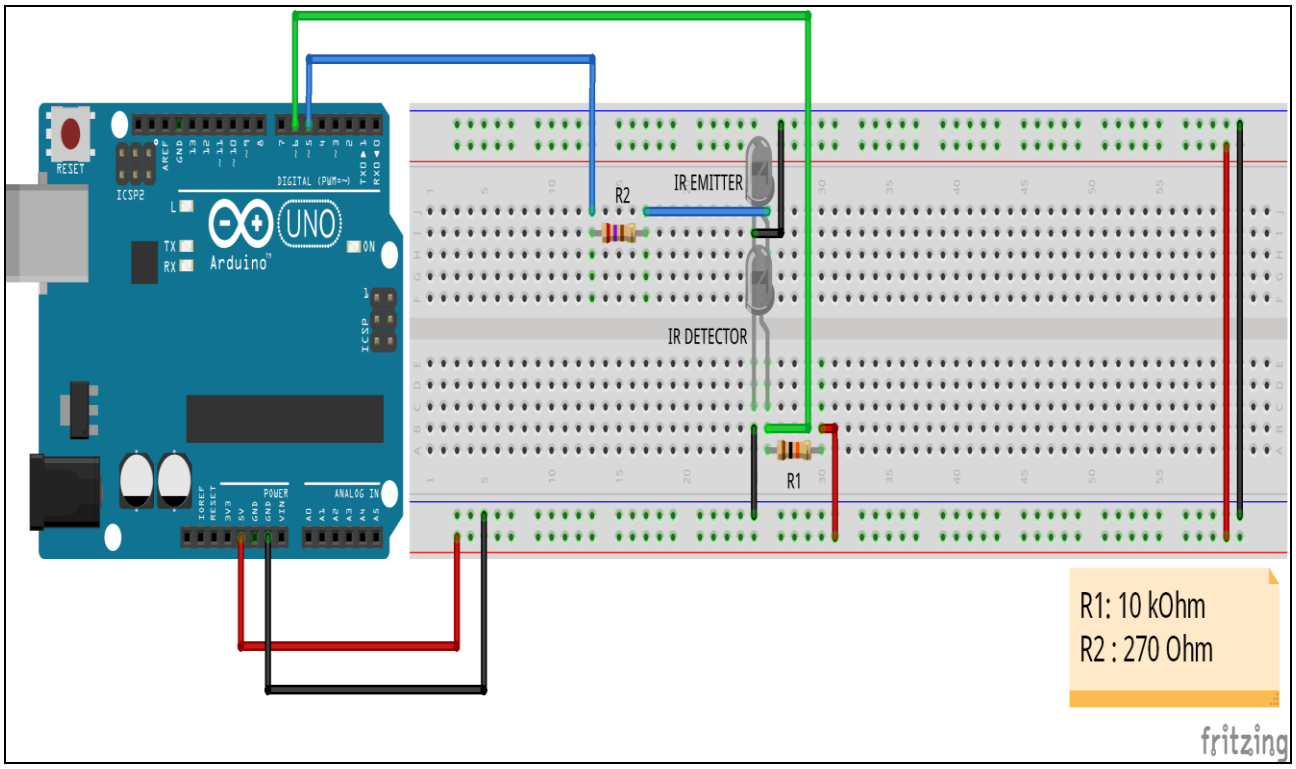
2.4.1 Gestione elettrovalvola e pompetta

Questo circuito elettrico serve per gestire il funzionamento della pompetta alimentata con 12V e l'elettrovalvola anche essa alimentata con 12V.



2.4.2 Gestione emettitore e trasmettitore a infrarossi

Questo circuito elettrico gestisce i due trasmettitori ad infrarossi (emitter e detector), tramite questi due si possono leggere le gocce o le righe che fa cadere l'elettrovalvola.



3 Implementazione

3.1 Creazione interfaccia web

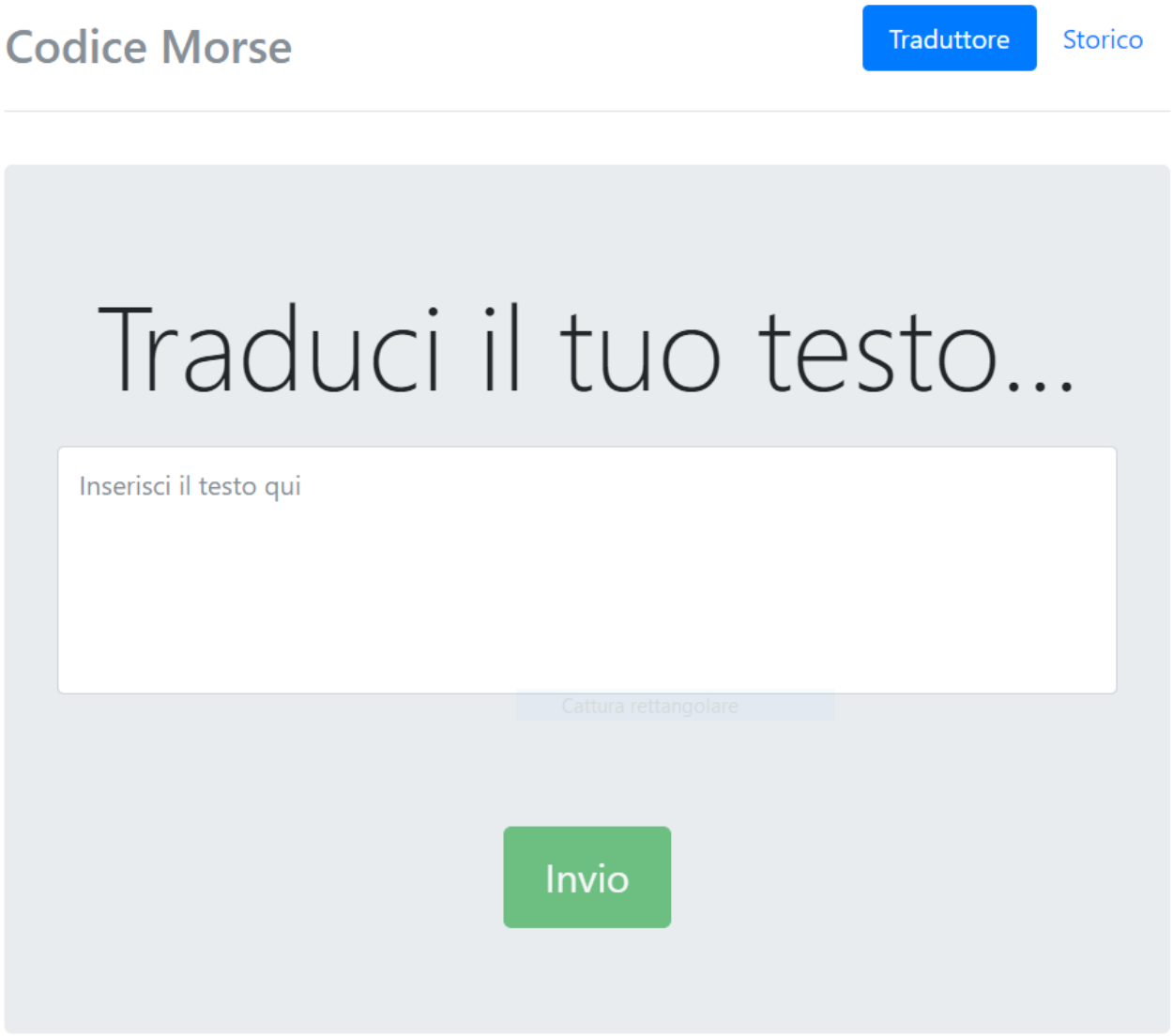
Come prima cosa ci siamo occupati di creare l'interfaccia del sito web basandoci sul design che abbiamo progettato in precedenza (pagina realizzata tramite il software Bootstrap).

3.1.1 Index.php

Abbiamo scaricato un template dal sito ufficiale di Bootstrap che fosse bello da vedere e con un design pulito, abbiamo scaricato il tema Narrow jumbotron. Dopodiché abbiamo cominciato la modifica del contenuto del template aggiungendo un form nel quale inserire il testo da tradurre ed un bottone per l'invio.

```
<form class="form" action="data_backup.php" method="post">

  <div class="form-group">
    <textarea maxlength="100" class="form-control" id="inputTextarea" name="inputTextarea"
    value="<?php if(isset($_POST['inputTextarea'])) echo $_POST['inputTextarea']; ?>"
    rows="3" placeholder="Inserisci il testo qui" required>
    </textarea>
  </div>

  <br>

  <p class="output" name="outputText" ></p>

  <br>

  <button name="checkButton" type="submit" class="btn btn-lg btn-success" disabled >Invio</button>
</form>
```

Le mofiche riportate alla pagina hanno portato al seguente risultato:



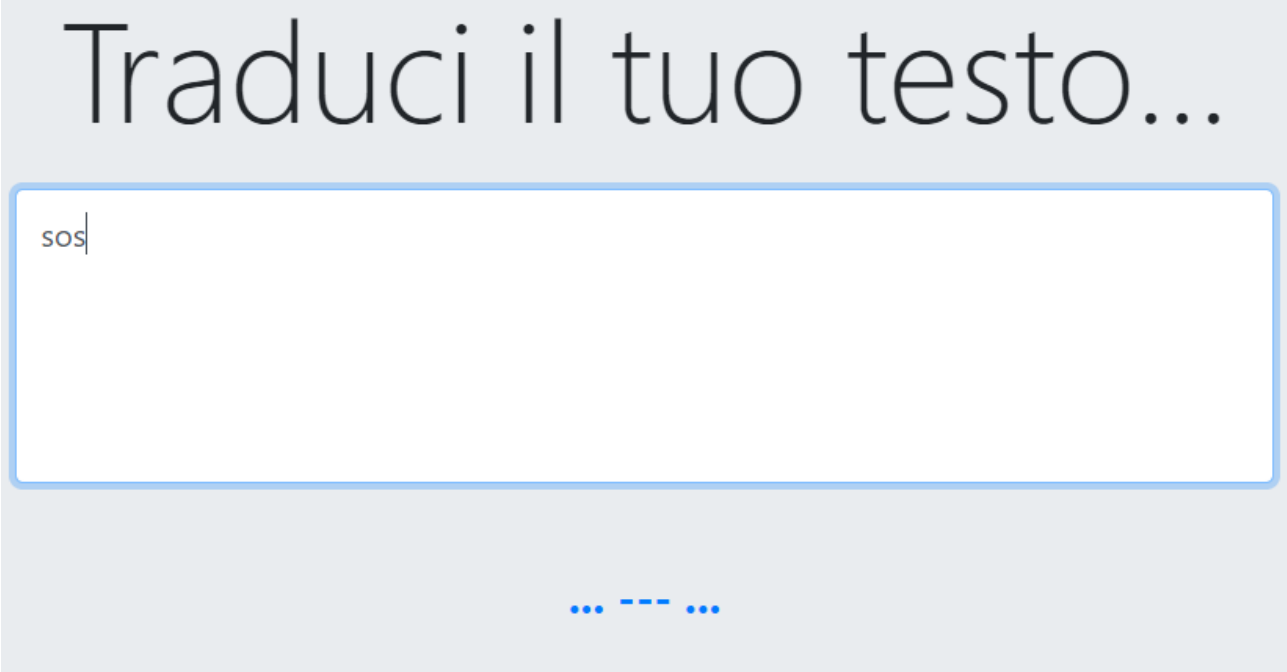
Cos'é il codice Morse?

Il codice Morse, detto anche alfabeto Morse, è un sistema per trasmettere lettere, numeri e segni di punteggiatura per mezzo di un segnale in codice ad intermittenza.

Cosa puoi codificare?

Il codice Morse codifica lettere alfabetiche e numeri, il nostro programma codifica le lettere (A-Z), i numeri (0-9). Gli spazi verranno codificati con '/' mentre i caratteri speciali con '#'.

All'interno della pagina index.php è presente uno script Javascript che rende il risultato della traduzione visibile dinamicamente all'utente.



3.1.2 Disable.js

Questo Javascript fa in modo che sulla pagina Web non sia possibile utilizzare il tasto destro e/o combinazioni di tasti per ispezionare la pagina.

```
/*
* Funzione che disabilita il tasto destro
*/
document.addEventListener("contextmenu", function(e){
  alert("This page is Protected by CPT, All Rights Reserved");
  e.preventDefault();
}, false);
```

3.1.3 Database

In seguito abbiamo creato il database rispettando il design che avevamo scelto all'inizio del progetto:

```
create database morsecode;
use morsecode;
create table code (id int primary key auto_increment,
encoded text not null, decoded varchar(255) not null);
```

3.1.4 Db_connection.php

Questo script serve per creare la connessione con il server, è stato creato per non dover essere ripetuto ogni volta che deve esserci una connessione.

```php
<?php
 //Connessione al db mysql
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "morsecode";
    $port = 3306;

    // Creo la connessione
    $conn = new mysqli($servername, $username, $password, $dbname, $port);
    // Controllo la connessione
    if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
    }
?>
```

Per poter richiamare questo script all'interno di altre pagine é possibile usare la funzione "require_once('db_connection.php');"

3.1.5 Data_backup.php

All'interno di questa pagine vengono eseguiti tre funzioni, la prima è quella di tradurre il teso in codice morse dato che il contenuto del form che arriva alla pagina è in testo non codificato. La logica è la medesima presente nell'index.php per tradurre il testo in codice morse.

```php
//Ciclo for scorre il testo
for ($i=0; $i < strlen($input); $i++) {
  $special = true;
  for ($y=0; $y < count($morse[0]); $y++) {
    //Se il testo é alfabetico o numerico aggiunge il morse
    if (strtoupper($input[$i]) == $morse[0][$y]) {
      $result .= $morse[1][$y]." ";
      $special = false;
    }
  }
  //Se non é un carattere alfabetico o numerico aggiungi #
  if ($special) {
    $result .= "#";
  }
}
```

La seconda funzione è quella di salvare il testo codificato e il testo non codificato all'interno del database.

```php
//Aggiungo i valori
 $sql = "INSERT INTO code (encoded, decoded)
 VALUES ('$result', '$input')";

 //Stampo il risultato dell'aggiunta dei valori
 if ($conn->query($sql) === TRUE) {
   echo "New record created successfully";
 } else {
   echo "Error: " . $sql . "<br>" . $conn->error;
 }

 $conn->close();
```

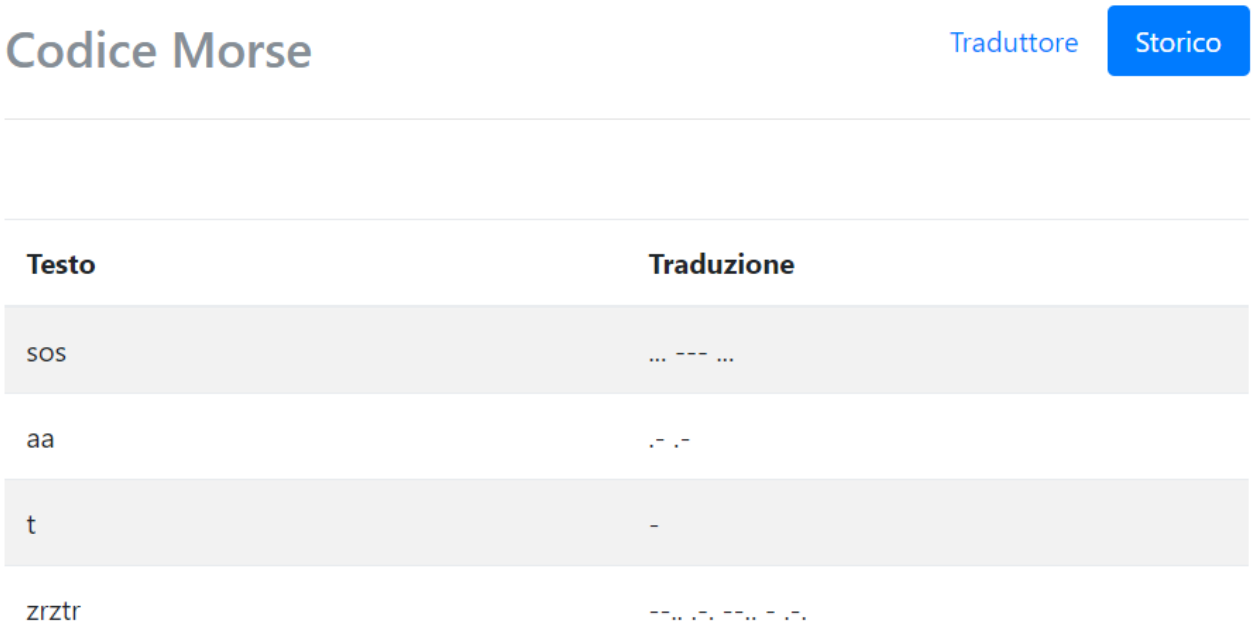
La terza ed ultima funzion é quella di inviare il testo codificato ad arduino, per poter fare ciò abbiamo usato una connessione UDP tramite i socket. Bisogna cambiare la configurazione di php.ini e abilitare i socket. Dal testo codificato vengono rimossi i simboli "/" e "#" che identificano gli spazi e i caratteri speciali, inoltre alla fine testo viene aggiunta una "E" che ci servirà nel programma di arduino per sapere quando il messaggio è terminato. Ad arduino viene inviato un carattere alla volta con una pausa di cento millisecondi tra uno e l'altro.

```php
$result .= "E";

$arduino_ip   = '192.168.20.2';
$arduino_port = 8888;
$mess = "";
$message = str_replace('#','',$result);
$message = str_replace(' ','*',$message);


if ($socket = socket_create ( AF_INET , SOCK_DGRAM , SOL_UDP )) {
  for ($i=0; $i < strlen($message); $i++) {

    socket_sendto($socket, $message[$i], 1, 0, $arduino_ip, $arduino_port);
    echo "<br>".$message[$i];

  usleep(10000);
  }
}else{
  print("can't create socketn ");
}
```

Dopodiché viene richiamata la pagina index.php in modo che l'utente non si accorga del passaggio di pagina.

3.1.6 History.php

Abbiamo creato una seconda pagina sempre utilizzando il medesimo template (Narrow jumbotron) nella quale si possono vedere in ordine cronologico dalla più recente (in alto) alla meno recente (in basso) e rispettando il design prestabilito all'inizio del progetto.



La pagina va a prendere tutti I valori presenti nel database e li stampa ogni volta che viene caricata.
Nel seguente script php è presente il codice che prende i dati dal db e li stampa nella pagina.

```php
<?php

require_once('db_connection.php');

//Prendo i valori dal db
$sql = "SELECT encoded, decoded FROM code ORDER BY id DESC";
$result = $conn->query($sql);

//Se esistono valori
if ($result->num_rows > 0) {
  //Stampo i valori di ogni riga in una tabella
  while($row = $result->fetch_assoc()) {
    echo "<tr>
    <td>".htmlspecialchars($row["decoded"])."</td>
    <td>".htmlspecialchars($row["encoded"])."</td>
    </tr>";
  }
}

//Chiudo la connessione
$conn->close();
?>
```

3.1.7 Valvola.ino

Questo programma è quello che gestisce l'elettrovalvola la quale fa scendere l'acqua quando è aperta e nulla quando è chiusa. Come arduino è stato necessario utilizzare uno con una entrata ethernet in modo che possa esserci la connessione coi socket con la nostra pagina data_backup.php, noi abbiamo scelto di utilizzare Arduino Leonardo ETH.
Inizialmente abbiamo dovuto trovare le librerie giuste per poter ricevere i dati via UDP.

```
#include <SPI.h>
#include <Ethernet2.h>
#include <EthernetUdp2.h>
```

Dopodiché é stato necessario settare tutte le variabili e costanti.

```
byte mac[] = {0x90, 0xA2, 0xDA, 0x10, 0xE2, 0x45}; //MAC dell'Arduino ethenet
IPAddress ip(192, 168, 20, 2); //IP dell'Arduino ethenet
unsigned int localPort = 8888; //Porta in ascolto dell'Arduino ethenet
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];// Lunghezza massima del Buffered
String code = ""; //Stringa code che conterrà il
const int mosfetPin = 7;
const int pausa = 300;
```

In seguito all'interno del metodo loop() è presente un controllo per verificare se sono arrivati dei dati "if (packetSize)" il quale controlla se i pacchetti ricevuti sono 0 o 1. Quando entra all'interno dell'if la memoria del buffered viene cancellata poiché il testo è stato salvato all'inetrno di un'altra variabile.

```
// read the packet into packetBufffer
Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);


Serial.println(packetBuffer);
code += packetBuffer;

//Resetto buffer
for(int i = 0; i < UDP_TX_PACKET_MAX_SIZE; i++) packetBuffer[i] = 0;
```

Una volta che la variabile contiene tutto il testo entra nel secondo if che controlla se l'ultimo carattere del testo contiene "E" in modo da sapere che il messaggio è completo, allora comincia ad aprire e chiudere la valvola con le tempistiche giuste per i punti, linee, e spazi. All'inizio e alla fine del messaggio viene mandato un protocollo per poter poi eseguire la lettura del messaggio.

```
if(code.charAt(code.length()-1) == 'E'){
  startStop();
  for(int i = 0; i < code.length(); i++){
    switch(code.charAt(i)){
      case '.':
        digitalWrite(mosfetPin, HIGH);
        delay(pausa);
        break;
      case '-':
        digitalWrite(mosfetPin, HIGH);
        delay(pausa*3);
        break;
      case '/':
        delay(pausa*7);
        break;
      default:
        delay(pausa);
        break;
    }
    digitalWrite(mosfetPin, LOW);
    delay(pausa);
  }
  startStop();
  code = "";
}
```

Il protocollo fa scendere due secondi di acqua continua all'inizio e alla fine del codice.

3.2 Lettura gocce con sensori IR

La lettura delle gocce viene fatta attraverso un emettitore a infrarossi e un lettore a infrarossi, ogni volta che qualcosa interrompe la frequenza passando in mezzo ai due la lettura ritorna 0.
Dopo aver letto il codice ricevuto lo traduce in italiano e lo stampa su uno schermo LCD.

3.2.1 Setup del codice che attiva il clock interno di Arduino

```
void setup() {
  pinMode(emitter, OUTPUT);
  pinMode(detector, INPUT);
  Serial.begin(9600);

  //inizialize Timer
  noInterrupts();           // Disabilito tutti gli interrupts
  TCCR1A = 0;               // TCCR1A, costante già esistente in Arduino
  TCCR1B = 0;               // TCCR1B, costante già esistente in Arduino
  timer1_counter = 65534;   // timer1_counter, variabile che rappresenta la frequenza
  TCNT1 = timer1_counter;   // TCNT1, costante già esistente in Arduino
  TCCR1B |= (1 << CS12);
  TIMSK1 |= (1 << TOIE1);
  interrupts();             // Riattivo tutti gli interrupts
}
```

3.2.2 Impostazione e memorizzazione di frequenza dell'emettitore

```
// Frequenza dell'emitter
bool frequencyE;
// Frequenza del detector
bool frequencyD;
ISR(TIMER1_OVF_vect) {
  TCNT1 = timer1_counter;
  // Imposto la frequenza dell'emitter IR
  digitalWrite(emitter, digitalRead(emitter) ^ 1);
  // EMITTER FREQUENCY
  frequencyE = (digitalRead(emitter) ^ 1);
  // DETECTOR FREQUENCY
  frequencyD = digitalRead(detector);
}
```

3.2.3 Attesa del protocollo dal'Ardiuno Ethernet e lettura

```
// ##########################################################################################
// Metodo per l'attesa del messaggio dall'Arduino Ethernet
// ##########################################################################################
void protocollo () {
  Serial.println("INIZIO PROTOCOLLO");
  bool esc = true;
  long timeOld = millis();
  long timeNew = NULL;
  bool readDetector = NULL;
  while(esc) {
    //Serial.println(digitalRead(6));
    if (digitalRead(6) == true) { // Passa l'acqua
      while(true) {
        Serial.println(digitalRead(6)); // DA NON CANCELLARE O COMMENTARE !!!!!
        if (digitalRead(6) == false) { // Non passa più l'acqua, interrompo il loop
          timeNew = (millis() - timeOld);
          esc = false;
          break;
        }
      }
    }
      if (timeNew >= timeOfProtocol) {
        Serial.println("PROTOCOLLO ESEGUITO");
        ascoltoMorse();
      }
  }
}
```

3.2.4 Ascolto costante dell'IR e memorizzazione del codice morse

3.2.4.1 Variabili

```
//###-VARIABILI-#########################################################################
String character = "";              // CARATTERE ALFABETICO ESPRESSO IN MORSE
String words[255];                  // CARATTERI DA TRADURRE
int index = 0;                      // INDICE PER L'ARRAY WORDS
int drop = 0;                       // COUNTER DELLE GOCCE E DELLE PAUSE
int oldCheck = 50;                  // VECCHIO VALORE
int newCheck;                       // NUOVO VALORE
//#######################################################################################
```

3.2.4.2 Se il valore precedente era acqua

```
if (oldCheck == 1) {                          // SE IL VALORE ERA ACQUA
  if (drop >= 0 && drop < 50) {
    Serial.println("DOT");
    character +=".";
  }
  else if (drop >= 50 && drop < 150) {
    Serial.println("DASH");
    character +="-";
  }
  else {
    Serial.println("END MESSAGE");
    translate(words);                         // QUANDO IL MESSAGGIO FINISCE FACCIO LA TRADUZIONE
  }
}
```

3.2.4.3 Se il valore precedente non era acqua

```
else {                                        // SE IL VALORE ERA PAUSA
  if (drop >= 0 && drop < 50) {
    // NIENTE
  }
  else if (drop >= 50 && drop < 150) {
    Serial.println("CHAR PAUSE");
    words[index] = character;
    index++;
  }
  else {
    Serial.println("WORD PAUSE");
    words[index] = " ";
    index++;
  }
}
```

3.2.5 Decodifica del messaggio

```
// ##############################################################################################
// Metodo che traduce il messaggio in codice morse ricevuto in una stringa in italiano
// ##############################################################################################
void translate(String str[]) {
  String result = "";
  String morse[] = {
                   ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..",
                   ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.",
                   "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--..",".----",
                   "..---","...--","....-",".....","-....","--...","---..","----.",
                   "-----", "/"
                   };
  char alfa[]={
              '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',' '
              };
  for (int i = 0; i < 255; i++) {
    for(int j = 0; j < 37; j++) {
      if(str[i] == morse[j]) {
        result += alfa[j];
      }
    }
  }
  return result; // STAMPA IL RISULTATO IN ITALIANO SUL LCD
}
```

3.3 Gestione pompetta e sensore a livello

La gestione della pompetta viene fatta attraverso un sensore a livello che controlla costantemente il livello dell'acqua e in base a quello decide se attivare o meno la pompetta.

3.3.1 Attivazione pompetta tramite sensore

Il seguente codice serve ad attivare la pompa utilizzando un sensore di umidità:

```
void loop(){

  int sensor = analogRead(A1);  //Legge il valore dato dal sensore
  Serial.println(sensor);

  if(sensor<200){               //Se il sensore legge un livello insufficente di acqua

    digitalWrite(3, HIGH);      //Fa partire la pompa

    delay(5000);                //Per una durata di 5 secondi

  }else{                        //Se il livello d'acqua è sufficiente

    digitalWrite(3, LOW);       //Spegne la pompa

  }
}
```

Il codice seguente invece è il setup del codice di Arduino per far funzionare il display LCD:

```
  lcd.begin(16, 2);         //Inizializza il display lcd
  lcd.setBacklight(255);    //Accende la luce del display
```

Questo invece è il metodo loop del codice di arduino per far funzionare il display LCD:

```
  lcd.print("Ciao!!!");     //Stampa sul display la stringa "Ciao!!!"
  delay(2000);
  lcd.clear();              //Ripulisce il display
```

4 Test

4.1 Protocollo di test

Test Case: **Riferimento**:	TC-001 REQ-001	**Nome:**	Realizzo un applicativo che possa comunicare in codice Morse
Descrizione:	Viene controllato che il sito offre delle pagine pubbliche per inviare delle frasi codificate in codice Morse ad una struttura comandata da due Arduini ma anche l'immagazzinamento delle frasi codificate all'interno di un database.		
Prerequisiti:	Bisogna avere accesso ad un software in cui é presente Apache con PHP e MySQL (es.: XAMPP).		
Procedura:	1. Accedere alla pagina in locale tramite il percorso *localhost/index.php* avente XAMPP accesso e vedere se la pagina è accessibile. 2. Testare la pagina index.php scrivendo la frase desiderata senza cliccare il bottone invia. 3. Testare la pagina index.php scrivendo la frase desiderata cliccando il bottone invia. 4. Testare la pagina di history.php che sia accessibile tramite l'index principale e al suo interno siano presenti tutte le frasi inserite in precedenza codificate e non.		
Risultati attesi:	1. La pagina sia raggiungibile tramite il percorso inserito in precedenza. 2. Scrivendo solo il testo senza premere il bottone mi aspetto che sotto il riquadro del testo appaia la codifica del testo soprastante. 3. Premendo il bottone invia non ricevo nessun tipo di errore. 4. Siano presenti tutte le frasi inserite e anche le rispettive codifiche.		

Test Case: **Riferimento**:	TC-002 REQ-002	**Nome:**	Realizzare un sito che permetta la codifica e la trasmissione del codice Morse
Descrizione:	Realizzare delle pagine che permettano di codificare e di visualizzare tutti i testi codificati e non codificati		
Prerequisiti:	Bisogna avere XAMPP con Apache e MySQL attivi, index.php.		
Procedura:	1. Accedendo alla pagina *localhost/index.php* deve essere possibile inserire del testo nell'apposito spazio. 2. Sotto lo spazio per l'inserimento del testo sarà presente lo spazio per la codifica. 3. Deve essere possibile cliccare il bottone invia senza problemi.		
Risultati attesi:	1. Una volta cliccato all'interno dell'apposito riquadro l'utente può scrivere tranquillamente senza alcun tipo di errore. 2. Appena l'utente scrive una lettera, sotto il riquadro per il testo apparirà la codifica di quella determinata lettera. 3. Una volta cliccato il bottone la pagina viene richiamata in modo tale che si svuoti.		

Test Case:	TC-003	**Nome:**	Realizzazione una seconda pagina web che rappresenterà lo storico del sito
Riferimento:	REQ-003		
Descrizione:	Realizzare una pagina che permetta la visualizzazione ordinata di tutte le frasi inserite nel corso del tempo con le loro rispettive codifiche.		
Prerequisiti:	Bisogna avere XAMPP con Apache e MySQL attivi, index.php.		
Procedura:	1. Tramite la pagina index.php deve essere possibile accedere allo storico (history.php). 2. Il sito permette la visualizzazione dello storico della pagina principale.		
Risultati attesi:	1. Dalla pagina index.php cliccando sulla voce *Storico* deve visualizzare la pagina *history.php.* 2. Al centro della pagina è presente la visualizzazione tramite tabella di tutti i testi e le rispettive codifiche inserite nella pagina principale.		

Test Case:	TC-004	**Nome:**	Interfaccia grafica del sito
Riferimento:	REQ-004		
Descrizione:	La grafica del sito deve rispettare lo scopo del progetto e oltretutto deve essere di facile intuizione per l'utente.		
Prerequisiti:	Avere XAMPP attivo		
Procedura:	1. Eseguire il collegamento al sito. 2. Visualizzare il campo d'input per il testo, il campo per la codifica e il bottone invia. In più l'accesso alla pagina dello storico.		
Risultati attesi:	Ci si aspetta che l'utente riesca a muoversi facilmente all'interno del sito.		

Test Case:	TC-005	**Nome:**	Gestione lettura codifica testo web-Arduino
Riferimento:	REQ-005		
Descrizione:	La lettura del testo codificato inviato dalla pagina web all'Arduino		
Prerequisiti:	Avere la pagina web principale per poter scrivere e inviare il testo codificato all'Arduino, XAMPP attivo, Arduino Ethernet con il relativo codice Arduino per ricevere la codifica e un cavo ethernet per collegare l'Arduino e il computer.		
Procedura:	1. Eseguire il collegamento al sito. 2. Eseguire il codice in Arduino e aprire il suo terminale. 3. Immettere del testo nella pagina web e cliccare Invio.		
Risultati attesi:	1. Il collegamento al sito avviene senza alcun tipo di problema. 2. Il codice si esegue senza alcun tipo di problema. 3. Immettendo del testo nell'apposito spazio all'interno della pagina web e cliccando invio sul terminale di Arduino ricevo i pacchetti di dati dalla pagina web.		

Test Case:	TC-006	**Nome:**	Gestione elettrovalvola.
Riferimento:	REQ-005		
Descrizione:	La gestione dell'acqua che l'elettrovalvola lascia cadere (gocce e "righe") in base da quello che riceve dalla pagina web.		
Prerequisiti:	Avere il circuito montato su breadboard o su placca che gestisce l'elettrovalvola più il relativo codice Arduino.		
Procedura:	1. Collegare il circuito all'elettrovalvola. 2. Eseguire il codice Arduino che gestisce l'elettrovalvola.		
Risultati attesi:	1. L'elettrovalvola fa cadere l'acqua ad intermittenza (gocce e "righe").		

Test Case:	TC-007	**Nome:**	Gestione emettitore e ricevitore infrarossi
Riferimento:	REQ-005		
Descrizione:	La lettura delle gocce e delle righe d'acqua rilasciate dall'elettrovalvola.		
Prerequisiti:	Avere l'elettrovalvola funzionante, un emettitore e un ricevitore a infrarossi con il loro rispettivo montaggio e codice.		
Procedura:	1. Collegare il circuito all'emettitore e al ricevitore a infrarossi. 2. Eseguire il codice Arduino per gestire i due sensori.		
Risultati attesi:	1. I due sensori riescono ad interpretare la quantità d'acqua che l'elettrovalvola lascia cadere.		

Test Case:	TC-008	**Nome:**	Gestione decodifica e stampa sul display LCD
Riferimento:	REQ-005		
Descrizione:	La decodifica del testo codificato e la stampa del testo originale sul display LCD.		
Prerequisiti:	Avere un Arduino, un display LCD e un codice per la decodifica e la stampa del testo.		
Procedura:	1. Collegare il display all'Arduino. 2. Avviare il codice Arduino per la gestione del display LCD.		
Risultati attesi:	1. Visualizzo il testo originale immesso nella pagina web sul display LCD.		

Test Case:	TC-009	**Nome:**	Gestione pompetta
Riferimento:	REQ-005		
Descrizione:	La pompetta che porta l'acqua dal recipiente posto sul fondo del tubo nel recipiente posto in cima al tubo.		
Prerequisiti:	Avere un Arduino, una pompetta con il rispettivo montaggio su breadboard o su placca e il rispettivo codice.		
Procedura:	1. Collegare la pompetta ai due recipienti tramite un tubicino. 2. Collegare l'elettrovalvola al rispettivo montaggio. 3. Avviare il rispettivo codice in Arduino.		
Risultati attesi:	1. In base al livello dell'acqua la pompetta si aziona e riempie il recipiente in cima fino ad un livello d'acqua sufficiente.		

Test Case:	TC-010	**Nome:**	Database
Riferimento:	REQ-006		
Descrizione:	Nel database esterno saranno presenti tutte le frasi scritte nella pagina principale.		
Prerequisiti:	Avere XAMPP con Apache e MySQL attivi, HeidiSQL, index.php e dataBackup.php.		
Procedura:	1. Accesso tramite HeidiSQL al database. 2. Visualizzazione di tutti i dati mandati dalla pagina principale.		
Risultati attesi:	1. Immettendo le credenziali di accesso al database si accede senza problemi al DB così da poter visualizzare la tabella dei dati. 2. Visualizzazione della tabella con tutti i dati mandati dalla pagina principale ogni volta che si premeva il tasto invia.		

4.2 Risultati test

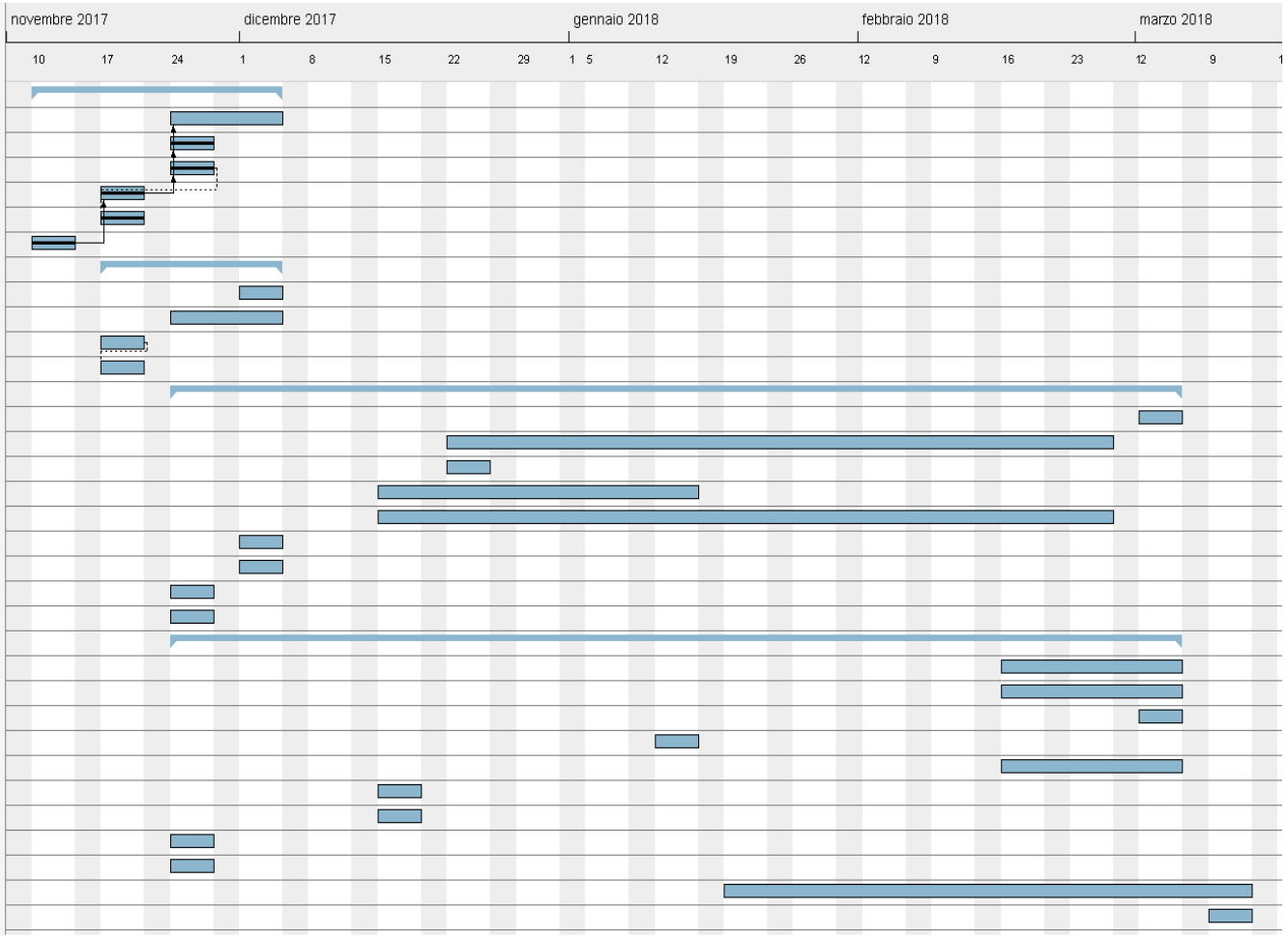
Test case	Nr. passaggio	Risultato
TC-001	1	La pagina è raggiungibile tramite il percorso.
	2	La codifica appare correttamente
	3	Cliccando il bottone non appaiono errori.
	4	All'interno dello storico sono presenti tutte le frasi inserite nella pagina principale.
TC-002	1	L'utente riesce a scrivere senza problemi.
	2	La codifica appare correttamente nell'apposito spazio.
	3	La pagina si resetta correttamente.
TC-003	1	Lo storico è accessibile correttamente.
	2	All'interno della tabella nello storico sono presenti tutti i testi con la codifica.
TC-004	1	Accesso al sito eseguito senza errori.
	2	Facile utilizzo della pagina web da parte dell'utente.
TC-005	1	Il collegamento al sito avviene senza problemi.
	2	Il codice si avvia senza dare problemi.
	3	Arduino riceve i pacchetti di dati inviati dalla pagina web.
TC-006	1	L'elettrovalvola lascia cadere delle gocce o delle righe in base al testo scritto.
TC-007	1	I due sensori non riesce a leggere le gocce o le righe d'acqua che scendono.
TC-008	1	Codice non implementato a causa delle tempistiche.
TC-009	1	La pompetta non funziona autonomamente perché non siamo riusciti ad implementare il sensore a livello che controlla il livello dell'acqua per problemi di tempistiche.
TC-010	1	Tramite le credenziali di accesso si effettua l'accesso senza problemi al DB.
	2	All'interno del DB sono presenti tutti i dati.

4.3 Mancanze conosciute

Prima della consegna finale del progetto abbiamo riscontrato due mancanze conosciute che riguardano quest'ultimo.
La prima mancanza riguarda il sensore a livello montato sul tappo in cima al tubo di plexiglass, quest'ultimo avrebbe dovuto tenere sotto controllo il livello dell'acqua presente nel recipiente posto in cima e, nel caso il livello si abbassasse, avrebbe dovuto azionare la pompetta per riportare l'acqua al livello adeguato. A causa di problemi riscontrati nella parte finale del progetto non siamo riusciti ad implementare questa parte.
La seconda mancanza riguarda la lettura della tempistica dell'acqua e della decodifica finale scritta sullo schermo LCD. Per la tempistica dell'acqua si intende un sensore posto un po' sotto la metà del tubo che avrebbe dovuto leggere l'acqua che usciva dall'elettrovalvola e avrebbe dovuto fare una distinzione tra gocce e "linee" d'acqua in modo tale da poter decodificare il testo e poterlo scrivere sullo schermo LCD. Però per colpa di problemi riscontrati, anche in questo caso, alla fine del progetto ci hanno costretto a risolvere questi ultimi portandoci ad avere dei problemi di tempistiche finali.

5 Consuntivo

Nome	Data d'i...	Data di fine
Analisi	10/11/17	01/12/17
analisi costi e benefici	24/11/17	01/12/17
analisi del dominio	24/11/17	24/11/17
analisi dei requisiti	24/11/17	24/11/17
risoluzione dei dubbi	17/11/17	17/11/17
docuento di specifica	17/11/17	17/11/17
intervista cliente	10/11/17	10/11/17
Progettazione	17/11/17	01/12/17
Design Database	01/12/17	01/12/17
Design struttura	24/11/17	01/12/17
Gantt	17/11/17	17/11/17
Design pagina web	17/11/17	17/11/17
Implementazione	24/11/17	02/03/18
Stampa su display il risultato finale	02/03/18	02/03/18
Ricezione e decodifica dati (Arduino...	22/12/17	23/02/18
Montare struttura	22/12/17	22/12/17
Ricezione e trasmissione dati (Ardui...	15/12/17	12/01/18
Lettura gocce tramite sensore	15/12/17	23/02/18
Controllo dati	01/12/17	01/12/17
Salvataggio nel file Database	01/12/17	01/12/17
Creazione pagina web	24/11/17	24/11/17
Codifica in Morse	24/11/17	24/11/17
Test	24/11/17	02/03/18
Test ricezione dati (Arduino UNO)	16/02/18	02/03/18
Test decodifica dati (Arduino UNO)	16/02/18	02/03/18
Test stampa su display	02/03/18	02/03/18
Test ricezione dati (Arduino Ethernet)	12/01/18	12/01/18
Test lettura goccie tramite sensore	16/02/18	02/03/18
Test codifica in Morse	15/12/17	15/12/17
Test salvataggio dati	15/12/17	15/12/17
Test performance	24/11/17	24/11/17
Test controllo dati	24/11/17	24/11/17
Documentazione	19/01/18	09/03/18
Consegna Progetto	09/03/18	09/03/18



5.1 Commento consuntivo

Abbiamo avuto difficoltà con la ricezione dei dati, questo ci ha rallentato di molto il progetto e per questo la stampa dei dati sul display è stata posticipata. Abbiamo riscontrato molti problemi nella lettura delle gocce tramite i sensori a infrarossi e questo ci ha costretto a posticipare la gestione della pompetta tramite il sensore a livello e anche i test finali.

6 Conclusioni

Il nostro progetto è quasi completo e le uniche cose mancanti sono la lettura delle gocce e la scrittura sullo schermo LCD della stringa decodificata, il resto è tutto funzionante, infine volevamo implementare l'attivazione della pompa tramite un sensore a livello ma per mancanza di tempo siamo riuscita a scrivere il codice e non l'abbiano testato.

6.1 Sviluppi futuri

Prima di tutto dovremmo finire di fare la lettura delle gocce rilasciate dall'elettrovalvola tramite i due sensori a infrarossi, ci sarebbe anche da implementare il sensore a livello per gestire in modo autonomo la pompetta e infine dobbiamo implementare la stringa decodificata stampandola sul display LCD.
.

6.2 Considerazioni personali

Per noi è stata una nuova esperienza lavorare in un gruppo con persone che conosciamo poco, è stato anche però facile coordinarsi nel gruppo perché durante la prima lezione ci siamo detti i nostri punti forti e deboli.
A nostro parere è stata una bella esperienza lavorare in gruppo perché ci ha fatto capire un po' di più l'importanza della collaborazione e anche ci ha permesso di capire come si lavorerà in una ditta.

7 Bibliografia

7.1 Sitografia

- https://stackoverflow.com/, Stack OverFlow, da 24.11.2017 al 19.01.2018.
- http://www.w3schools.com/, W3Schools, da 24.11.2017 al 19.01.2018.
- http://php.net/, PHP, da 24.11.2017 al 19.01.2018.
- https://www.arduino.cc/, Arduino, da 24.11.2017 al 19.01.2018.
- http://www.danielealberti.it/, Arduino 's blog, da 24.11.2017 al 19.01.2018.
- https://getbootstrap.com/, Bootstrap, da 24.11.2017 al 19.01.2018.
- https://github.com, github, da 24.11.2017 al 9.03.2018.

8 Allegati

Elenco degli allegati, esempio:

- Diari di lavoro
- Codici sorgente
- Mandato
- Presentazione