UNIVERSITA' DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA “ENZO FERRARI”

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

Prototipo per il monitoraggio della temperatura ed altezza dell’acqua di un pozzo mediante sistema a microcontrollore

Relatore:

Prof. Ing. Augusto Pieracci

Laureando:

Gozzi Davide

Anno Accademico 2016/2017

Indice

1. Introduzione

Il terremoto è un evento imprevedibile e distruttivo, i segni che manda prima dello scatenarsi sono difficilmente identificabili ed altrettanto difficile è cercare di omologarli e schedarli per provare a comprendere più a fondo il fenomeno ed avere un tempo maggiore prima dello scatenarsi del terremoto per organizzarsi e ridurre al minimo i danni.
Oltre alle classiche scosse premonitrici, sono presenti anche fenomeni secondari che sono poco studiati in quanto non sempre presenti durante lo sciame sismico: essi sono detti fenomeni secondari.

Un fenomeno secondario che è stato presente durante lo sciame sismico che ha colpito la provincia di Modena nell'estate del 2012, è stato l'aumento della temperatura dell'acqua del pozzo in maniera molto sensibile, portandola dai 15°C medi fino a raggiungere picchi superiori ai 45°C.

Per monitorare questo particolare fenomeno si è deciso di progettare un prototipo basato sul microcontrollore Arduino, il quale gestisce il rilevamento e la trasmissione dei dati relativi alla temperatura e all'altezza del pozzo, inviandoli a ThingSpeak, un sito internet dotato di database per la memorizzazione e la graficazione dei dati.

2. Terremoto

Le rocce fuse che formano l'interno della Terra non sono omogenee ma presentano zone con pressione, temperatura, densità e caratteristiche dei materiali molto diverse. Tale disomogeneità induce lo sviluppo di forze che tendono a riequilibrare il sistema fisico-chimico. Queste forze si manifestano nei moti convettivi. Essi determinano dei movimenti negli strati più superficiali della Terra, spingendo le masse rocciose le une contro le altre, deformandole. La Terra è dunque un sistema dinamico in continua evoluzione e i terremoti sono una conseguenza dei processi dinamici e tettonici che determinano la genesi e l'evoluzione dei bacini oceanici, delle catene montuose e dei continenti. Quando tali deformazioni raggiungono il limite di resistenza dei materiali, questi si fratturano liberando quasi istantaneamente l'energia elastica fino ad allora accumulata. L'energia si propaga in tutte le direzioni sotto forma di onde sismiche, provocando quei movimenti del suolo che costituiscono il terremoto.

Parte dell'energia rilasciata durante un terremoto prende la forma di onde sismiche; esse viaggiano attraverso la Terra producendo lo scuotimento del terreno anche a grandi distanze dalla sorgente del terremoto.

Le forza che agiscono sulla superficie terrestre producono nella crosta formazioni quali bacini oceanici, continenti e catene montuose. Queste forze vengono interpretate con la teoria della "Tettonica delle placche" che fornisce una spiegazione fisica razionale della maggior parte dei terremoti e dei processi geologici che avvengono sulla superficie terrestre. Il modello della tettonica a placche identifica una serie di placche litosferiche a comportamento rigido che si muovono determinando diversi tipi di margini nelle aree di contatto. La distribuzione spaziale dei terremoti segue i bordi delle placche disegnando delle fasce caratteristiche in cui si concentrano anche altri aspetti dell'attività della Terra. Queste fasce sono le zone di interazione tra le varie placche in movimento, sede di accumulo di grande energia che viene costantemente liberata nel corso dei terremoti che interessano la Terra. [1]

La rottura della roccia durante un terremoto è accompagnata dall'improvviso rilascio dell'energia accumulata che si propaga sotto forma di onde sismiche in tutte le direzioni. Il punto all'interno della terra dove si genera la rottura si chiama ipocentro del terremoto; il punto corrispondente sulla superficie terrestre è chiamato epicentro.

Una faglia è una frattura o un sistema di fratture lungo le quali la roccia risulta dislocata.
L'indicazione del tipo di terremoto è data da due parametri: l'intensità, che misura la grandezza di un terremoto attraverso l'osservazione dei danni e degli effetti del terremoto sull'uomo, sulle costruzioni e sull'ambiente; la magnitudo, che misura la forza di un terremoto in termini di energia rilasciata durante l'evento, attraverso le registrazioni degli strumenti come i sismografi.

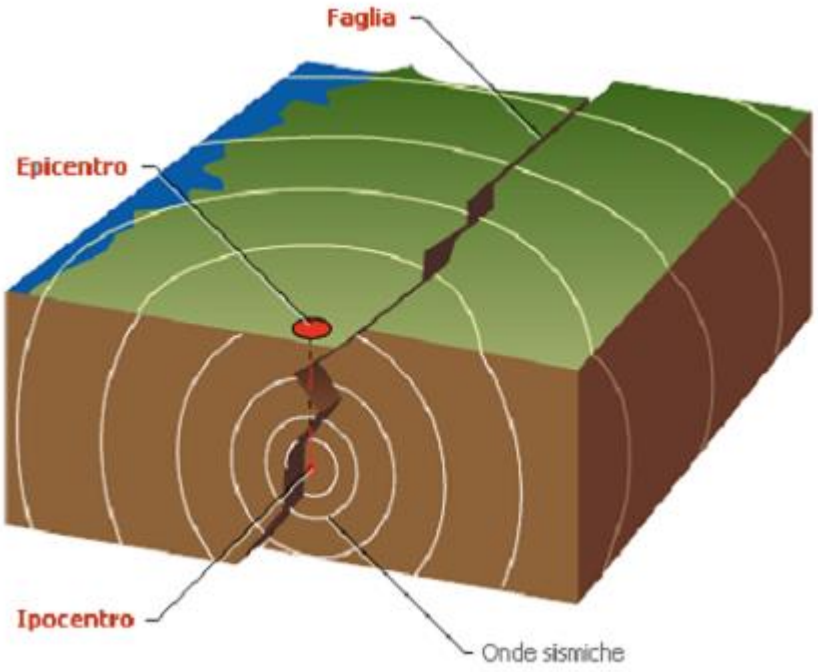

Fig. 1 - Schema di un terremoto

Le onde sismiche si propagano sfericamente a partire dall'ipocentro, ossia il punto all'interno della Terra da dove si sprigiona l'energia. Si distinguono tre tipi di onde sismiche:

- Onde longitudinali o di compressione (P): le onde P (onde primarie) fanno oscillare la roccia avanti e indietro, nella stessa direzione di propagazione dell'onda. Poiché le onde P si propagano più rapidamente, sono anche le prime a raggiungere i sismografi, e quindi ad essere registrate.
- Onde di taglio o trasversali (S): le onde S, ovvero onde "seconde", muovono la roccia perpendicolarmente alla loro direzione di propagazione (onde di taglio).
- Onde superficiali (R e L): le onde superficiali, a differenza di quello che qualcuno potrebbe pensare, non si manifestano dall'epicentro, ma solo ad una certa distanza da questo. Tali onde sono il frutto del combinarsi delle onde P e delle onde S e sono perciò molto complesse. Le onde superficiali sono quelle che provocano i maggiori danni.

Attualmente, per misurare i terremoti, si utilizzano due tipi di scale: scala Richter, la quale misura la Magnitudo o Energia del terremoto e Scala Mercalli che misura l'intensità del terremoto e si basa sugli effetti macrosismici.
La scala Richter, è stata introdotta nel 1935 dal sismologo C. Richter e serve a definire la magnitudo esprimendo l'energia liberata dal sisma.
Richter scelse arbitrariamente una magnitudo zero per un terremoto che mostri uno spostamento massimo di un micrometro (1/1000 di mm) sul sismografo di Wood-Anderson, se posto a 100 km di distanza dall'epicentro del terremoto, cioè più debole di quanto si potesse registrare all'epoca.
La scala Mercalli, invece, misura l'intensità del terremoto basandosi su effetti macrosismici (danni a persone e manufatti) ed è quindi una misura molto imprecisa, perché i danni rilevati dipendono anche dalle caratteristiche delle strutture, dalla densità abitativa, dall'importanza artistica di determinati edifici e da altre variabili indipendenti dal terremoto stesso.

I terremoti di maggiore magnitudo sono di norma accompagnati da altri eventi secondari, non necessariamente meno distruttivi, che seguono la scossa principale e si definiscono repliche, spesso definite in modo non corretto scosse di assestamento.

2.1 Pozzo ed eventi collegati al terremoto

Un pozzo è una struttura che raggiunge una determinata profondità del terreno, di norma non superiore ad alcuni metri, ed è in grado di far emergere l'acqua che è naturalmente contenuta all'interno delle falde acquifere.
Si distinguono due tipologie di falde acquifere: le falde freatiche e le falde artesiane.

Le falde freatiche sono collocate a poca profondità, quindi l'acqua può essere ricavata facilmente, per contro tuttavia, non si ha la certezza che l'acqua sia pura, proprio perchè potenzialmente potrebbe essere contaminata dal terreno.

Le falde artesiane, quelle da cui prelevano l'acqua i pozzi artesiani, sono invece delle falde acquifere in cui l'acqua è contenuta tra due strati di terreno impermeabile, dunque il rischio di contaminazioni dell'acqua è notevolmente inferiore.

Le falde acquifere possono essere alterate da eventi sismici, infatti oltre alle repliche, durante il periodo che precede e segue un terremoto possono verificarsi fenomeni poco noti e quindi poco studiati che si possono manifestare all'interno di un pozzo. Questa classe di fenomeni geologici di

breve durata e con bassa frequenza di accadimento comprende vari fenomeni tra cui la emissione improvvisa di gas o acque dal sottosuolo, la formazione di cedimenti della superficie, la formazione di vulcanetti di sabbia, l'accadimento di fenomeni luminosi nell'atmosfera, l'intorbidimento, la variazione dell'altezza e il surriscaldamento dell'acqua dei pozzi. Molti fenomeni di questa natura occorsi nel territorio Italiano o nel mondo antico sono stati studiati e riferiti anche nella letteratura scientifica storica o antica come conseguenze dei terremoti. [2]

Già nel 1789 con De Rossi e nel 1883 con Mercalli non è raro trovare traccia di notizie della fuoriuscita di liquidi, anche caldi dal sottosuolo, oppure il cambiamento della temperatura e della composizione dell'acqua delle sorgenti o dei pozzi.
Anche Sorbelli nel 1910 descrivendo il terremoto avvenuto in Molise nel 1456 riferisce l'intorbidimento e l'aumento di temperatura dell'acqua dei pozzi:" El primo quando l'acqua del pozzo vene torbida, e questo adivene per gli venti che voglino uscire fuori de la terra " (Sorbelli, 1910). [3]
Questi fenomeni sono stati osservati anche durante il terremoto in Emilia Romagna del 2012, infatti durante la sequenza sismica e nel periodo successivo si è registrato un deciso incremento di anomalie termiche nell'acqua di alcuni pozzi. Si è trattato generalmente di pozzi freatici, profondi una decina di metri circa, in cui la temperatura è a volte salita anche oltre i 40 °C, a fronte di una temperatura che solitamente è attorno ai 15 °C.

Fig. 2 – Pozzo dopo sciame sismico Emilia 2012

Un evento similare si è verificato in un pozzo di Medolla (MO) alcuni giorni prima del 22 Ottobre 2015, giorno in cui ha avuto luogo un terremoto di magnitudo 3.5 della scala Richter, in cui l'acqua di pozzo ha raggiunto temperature superiori ai 40°C.

Si è perciò deciso, per monitorare questo particolare fenomeno, di progettare un prototipo basato sul microcontrollore Arduino, il quale gestisce il rilevamento e la trasmissione dei dati relativi alla temperatura e all'altezza del pozzo, inviandoli a ThingSpeak, un sito internet dotato di database per la memorizzazione e la graficazione dei dati.

3. Componenti Hardware

In questo capitolo verranno trattati tutti i sensori e i dispositivi utilizzati nel progetto;
i collegamenti elettrici, visti sulla breadboard, sono visibili nella foto sottostante.
L'immagine (Fig. 3) è stata realizzata grazie al programma Fritzing.

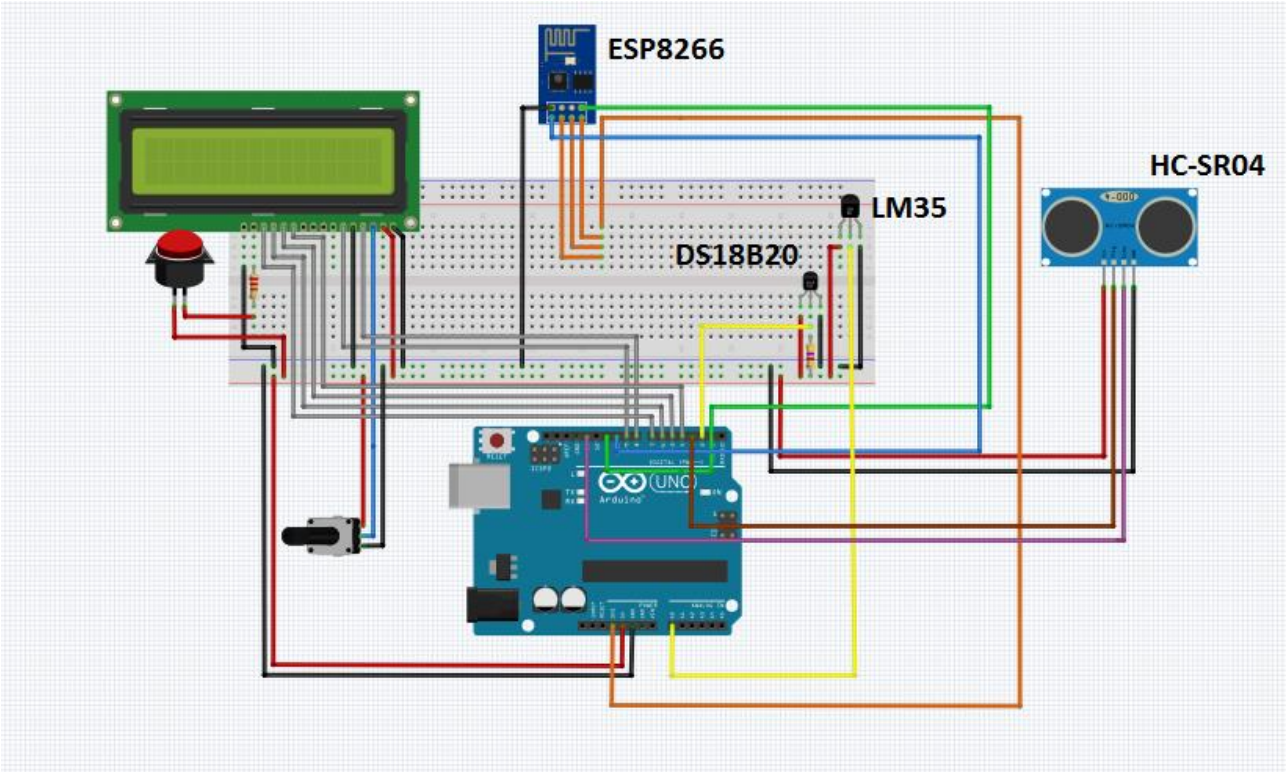

Fig. 3 – Schematico progetto

3.1 Arduino

Arduino Uno è sicuramente una delle schede più diffuse e utilizzate in tutto il mondo. E' una scheda con microcontrollore ATmega328. Dispone di 14 pin digitali di ingresso/uscita, sei dei quali possono essere usati come output PWM (Pulse-width modulation), sei sono ingressi analogici; possiede inoltre un risonatore ceramico da 16 MHz, un connettore USB, un jack di alimentazione, un header ICSP4 ed un pulsante di reset. [4]
Contiene tutto il necessario per lavorare con il microcontrollore.

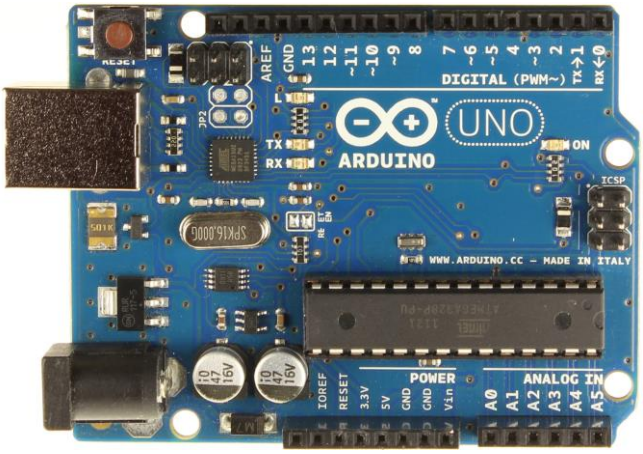

Fig. 4 – Arduino Uno

Una volta collegato ad un computer, ed installato l'ambiente di sviluppo, è possibile già da subito cominciare a programmare la scheda. Di seguito vengono descritti con maggior dettaglio alcune parti della scheda e nella figura è possibile vedere il fronte di Arduino Uno (Fig. 4). [5]

Il microcontrollore di Arduino Uno R3 è l'ATmega328, un microcontrollore ad alte prestazioni con bus a 8 bit prodotto dalla Atmel con architettura di tipo RISC (Reduced Instruction Set Computer) le cui caratteristiche sono elencate nel seguente elenco:

- Memoria di programma Flash da 32 kb
- Memoria EEPROM da 1 kb
- Memoria SRAM da 2 kb
- Prestazioni di 20 MIPS a 20 MHz
- 23 linee di I/O programmabili
- 32x8 registri general purpose
- Due timer a 8 bit e uno a 16 bit
- Sorgenti interne ed esterne per interrupt
- Seriale USART (Universal Synchronous/Asynchronous Receiver/Transmitter)
- Interfaccia Seriale I2C
- Porta Seriale SPI
- Sei convertitori A/D con risoluzione 10 bit
- Tensione di funzionamento compresa tra 1.8V e 5.5V
- Temperatura di lavoro da -40°C a +85°C
- Basso consumo di corrente

Le porte general purpose sono collegate sulla scheda in maniera da essere utilizzate come porte digitali o analogiche.

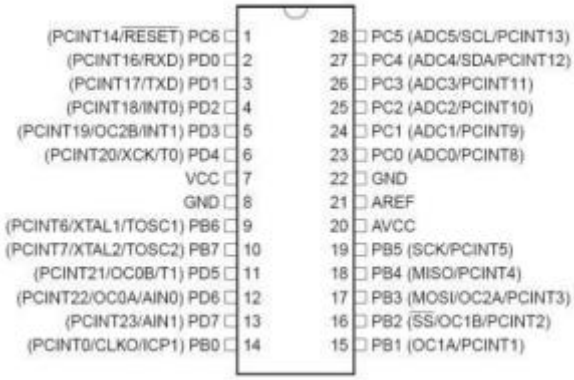

Fig. 5 – Porte Arduino Uno

Il microcontrollore ATmega328 dispone di 32 kb di memoria di programma della quale 0,5 kb sono usati per il bootloader.
L'Arduino Uno può essere alimentata via USB, o da un'alimentazione esterna.
La sorgente di alimentazione viene selezionata automaticamente dalla scheda. L'alimentazione esterna può provenire sia da un adattatore AC/DC che da una batteria. L'adattatore può essere collegato ad Arduino tramite un jack da 2.1mm. La batteria può essere collegata ai pin GND e VIN sul connettore di alimentazione della scheda.
La tensione di alimentazione esterna deve essere compresa tra i 6V ed i 20V, anche se le tensioni raccomandate sono dai 7V ai 12V.

I pin di alimentazione sono disponibili tramite il connettore POWER e sono i seguenti:
VIN: restituisce la tensione applicata dall'alimentatore al jack e può essere usato per alimentare altri circuiti che dispongano già di un regolatore di tensione.
5V: fornisce i 5 volt prelevati dall'uscita del regolatore interno ed è utile per alimentare altri circuiti compatibili con i 5 volt.
3.3V: fornisce i 3,3 volt ricavati dal regolatore corrispondente e consente di alimentare altri circuiti compatibili con tensioni di 3,3 volt (la massima corrente prelevabile è di 150 mA).
GND: è il contatto di massa (Ground).

RESET: portando questa linea a livello basso permette di resettare il microcontrollore. Questa funzionalità è disponibile anche attraverso un tasto, chiamato appunto Reset.
IOREF: consente agli Shield di adattarsi alla tensione fornita dalla scheda.

Ciascuno dei 14 pin digitali presenti sulla Arduino Uno, possono essere configurati in modalità input oppure output. La tensione di uscita di ogni output è di 5 volt, ed ogni pin è in grado di lavorare con un massimo di 40 mA.
Ogni pin è dotato di una resistenza di pull-up del valore di 20-50 kΩ.

Alcuni pin hanno anche comportamenti specifici:
Pin 0 (RX) e 1 (TX): possono essere utilizzati per la comunicazione seriale.
Pin 2 e 3: possono essere configurati come trigger per eventi esterni, come ad esempio il rilevamento di un fronte di salita o di discesa di un segnale in ingresso. Le resistenze di pull-up e pull-down sono usate nei circuiti elettronici per forzare un determinato stato logico ad un determinato valore, per eliminare fluttuazioni di corrente o ancora per evitare cortocircuiti quando si usano i pulsanti.
Pin 3, 5, 6, 9, 10 e 11: possono essere configurati via software per generare segnali PWM con risoluzione di 8 bit. Tramite un filtro RC è possibile ottenere tensioni continue di valore variabile.
Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK): possono essere programmati per realizzare una comunicazione SPI.
Pin 13: è connesso a un LED interno alla scheda e risulta utile in situazioni di debug.
GND: è il contatto di massa (Ground).
AREF: tensione di riferimento per gli ingressi analogici.

Arduino Uno ha sei ingressi analogici etichettati da A0 ad A5, ognuno dei quali fornisce 10 bit di risoluzione e quindi 1024 valori differenti. Per impostazione predefinita possono misurare una tensione di 5 volt riferita a massa, anche se è possibile cambiare l'estremità superiore del loro intervallo utilizzando il pin AREF e la funzione *analogReference()*.

Alcuni hanno comportamenti specifici:
Pin A4 (SDA) e A5 (SCL): permettono di realizzare una comunicazione nello standard I2C a due fili.

3.2 Sensore HC-SR04

Il sensore scelto è stato l'HC-SR04 (Fig. 6). La scelta è ricaduta su questo sensore per diversi motivi: esso è economico, immediato da usare e molto diffuso.
Secondo i dati forniti dal datasheet ha una portata di circa 4m ed una risoluzione di circa 30mm.
Il sensore è dotato di 4 pin: due di essi sono Vdd e GND e servono per alimentarlo, gli altri due si chiamano TRIG ed ECHO. Il primo accetta in ingresso un impulso di una decina di µs a livello logico alto per inizializzare l'operazione di lettura della distanza. Il secondo restituisce un segnale a livello logico alto di durata proporzionale alla distanza: una durata di 58µs corrisponde ad 1cm. [6]

Fig. 6 – Sensore ad ultrasuoni HC-SR04

3.3 Sensore DS18B20

Il sensore DS18B20 è un sensore digitale di temperature con una risoluzione che varia dai 9 ai 12 bit per la misurazione di temperature in gradi Celsius, esso è allocato all'interno di un involucro a tenuta stagna, e anche se questa soluzione provoca un notevole innalzamento delle temperature nel caso di irraggiamento diretto, è perfetta per il progetto in quanto dovendolo immergere nell'acqua è richiesta impermeabilità. [7]

Fig. 7 – Sensore di temperatura ad immersione DS18B20

E' in grado di operare in un range di temperatura che va da -55°C ai +125°C con una risoluzione di 0.5°C nel range che va da -10°C a +55°C, quello più utilizzato nel progetto.
L'alimentazione può variare da 3V a 5.5V; nel progetto si è utilizzata l'alimentazione a 5V.
Il collegamento al circuito avviene attraverso tre fili corrispondenti ad alimentazione, massa ed il segnale DQ (Data IN), come utilizzato nel progetto. È prevista anche una modalità di funzionamento parassita ovvero prelevando la tensione di alimentazione direttamente dalla linea dati, riducendo a due il numero di collegamenti necessari. Ogni sensore ha un codice seriale a 64 bit univoco, in questo modo è possibile far funzionare più dispositivi sulla stessa linea di comunicazione. Il sistema risulta molto semplice e il microprocessore riesce a gestire diversi sensori anche su aree molto vaste. [8]
Il pin dell'alimentazione è collegato ad Arduino, il quale fornisce 5V, quello di massa alla massa, mentre il pin DQ andrà collegato ad Arduino e all'alimentazione tramite una resistenza da 4.7kΩ.
Durante la fase di cablaggio il cavo del sensore è stato allungato e dotato di un peso sulla punta per poter raggiungere il fondo del pozzo ed evitare che galleggi o venga spostato dal risucchio generato dal motorino per l'irrigazione che aspira l'acqua del pozzo.

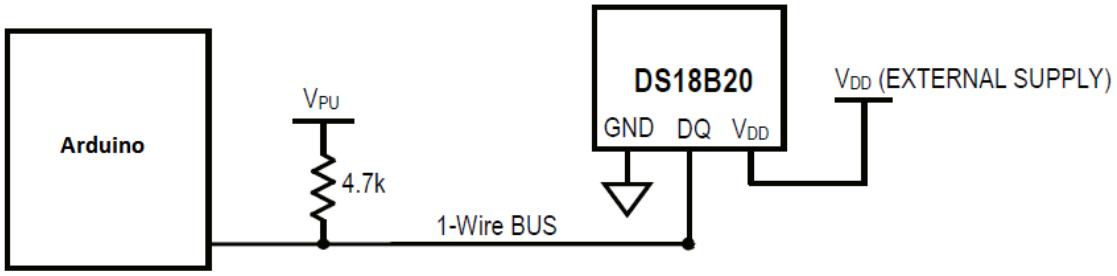

Fig. 8 – Collegamento DS18B20 con Arduino

3.4 Display Lcd

Questi display sono tipicamente costruiti con case diversi, ma posseggono tutti un aspetto che li accomuna: il controller utilizzato, basato sul diffusissimo HD44780 di Hitachi.
Grazie a questa caratteristica comune, il software implementato per interfacciare un particolare LCD è quasi completamente compatibile. Il controller dialoga con il microcontrollore mediante comandi definiti secondo parametri standard indicati sulle tabelle dei datasheet ma la configurazione dei pin dell'LCD potrebbe variare quando si cambia modello di display.

Nel progetto si è utilizzato, quindi, un classico display 16x2, il quale ha 2 righe composte da 16 caratteri ognuna.

Fig. 9 – Display Lcd

Molti display proprio come quello in esame sfruttano una alimentazione a 5V; la corrente che viene assorbita dal display, dipende in larga misura dall'estensione (in termini di caratteri) e dalla potenza necessaria per l'accensione dei singoli pixel. [9]
Viene generalmente fornita la possibilità di modificare la regolazione del contrasto dei caratteri con lo sfondo, nel nostro caso si è ottenuto mediante l'utilizzo di un trimmer collegato come in figura (Fig. 10) che permette una regolazione ottimale del contrasto desiderato. In alternativa è possibile fissare il livello di contrasto, sfruttando un partitore resistivo di valore noto, si evita così l'uso di un trimmer.

Il controller HD44780 consente l'interfacciamento con un microcontrollore mediante due modalità di connessione:
- Con bus dati a 8 bit;
- Con bus dati a 4 bit.

Le due alternative si distinguono sia per le diverse impostazioni software sia per il numero di pin impiegati. Entrambe le modalità hanno però in comune il numero di pin di controllo.
In particolare:

- E: L’ Enable è pin sul quale viene inviato il segnale di sincronismo. Quando viene posto a livello logico alto, il controller esegue la lettura del dato presente sulla linea dati, lo decodifica e successivamente esegue sul display il carattere o comando corrispondente.

- R/W: Read or Write, il display può funzionare in modalità sia di lettura che di scrittura, anche se normalmente viene sempre utilizzata la sola funzione di scrittura. In generale se il pin viene configurato a livello logico basso, si desidera scrivere un dato, viceversa se viene posto a livello alto si vuole leggere un dato;

- RS: Register Select, questo pin viene usato per comunicare al display il tipo di informazione che stiamo per inviare sul bus di dati; se RS viene posto a livello logico alto, si desidera inviare un carattere da visualizzare sul display, se invece lo poniamo a livello logico basso, stiamo inviando un comando da eseguire; esistono vari tipi di comandi, i più comuni sono: selezione della riga su cui scrivere, riposizionamento del cursore e il clearscreen.

La modalità con bus dati a 4 bit consente un risparmio dei pin del microcontrollore, senza far aumentare significativamente la complessità del controllo. Per questa ragione, nel progetto si questa modalità, visibile nella seguente figura.

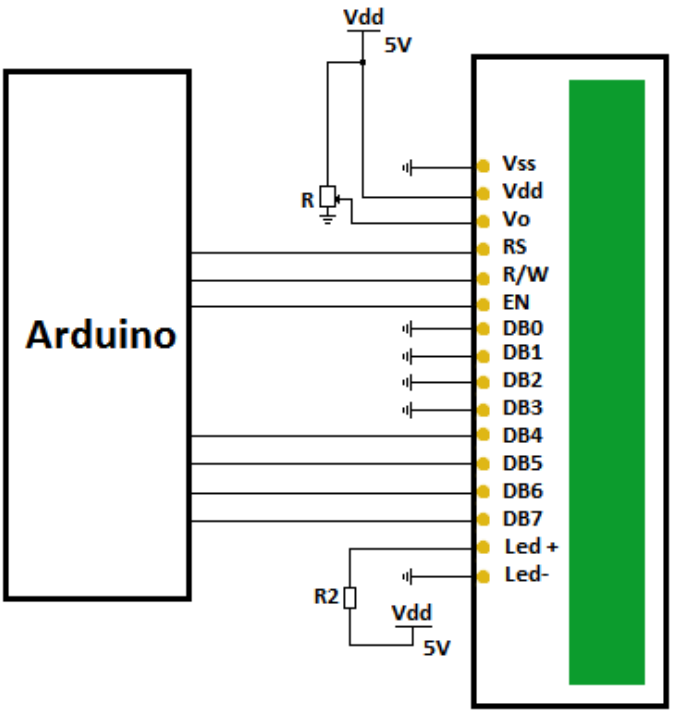

Fig. 10 – Collegamento Display ad Arduino

3.5 Sensore LM35

Il sensore integrato LM35 è un trasduttore di temperatura con una buona precisione, con la tensione d'uscita proporzionale alla temperatura. (Fig. 11)

Fig. 11 – Sensore di temperatura LM35

L'uscita proporzionale alla temperatura è espressa in gradi Celsius (Centigradi). Questo è un vantaggio rispetto ai sensori calibrati sulla scala di temperatura in gradi Kelvin, perché l'utilizzatore

non deve fare nessun tipo di operazione per passare dalla scala di temperatura in gradi Kelvin a quella in Celsius.
Il sensore LM35 non ha bisogno di nessun tipo di calibrazione esterna. Ha un'accuratezza pari a ±3/4°C nell'intervallo di temperatura che va da -55 a 150 °C. La sua bassa impedenza, la sua ottima linearità, la precisione della calibrazione, fanno sì che il sensore si interfacci ottimamente con i circuiti di lettura della sua tensione d'uscita. [10]
Esso può essere usato anche con diversi valori della tensione di alimentazione.
L'intervallo delle tensioni di alimentazione va da 4 a 30 V.
Il sensore LM35 ha tre terminali: uno da cui è fornita l'alimentazione, uno di uscita e un terzo per la massa.
La tensione d'uscita fornita è pari a:

$$Vout = K \times T$$

dove K è la sensibilità del sensore ed è pari a 10 mV/°C e T è la temperatura in gradi Celsius.
Quindi, ogni aumento di 1°C della temperatura del misurando, la tensione d'uscita aumenta di 10mV.
La corrente utilizzata dal sensore cambia in base alla tensione di alimentazione utilizzata, rimanendo comunque sotto i 60µA per qualunque tensione si usi come alimentazione.

3.6 ESP8266

Il modulo WiFi ESP8266 (Fig. 12) è un sistema a circuito integrato che permette di connettere se stesso e qualsiasi altro microcontrollore alla rete WiFi.

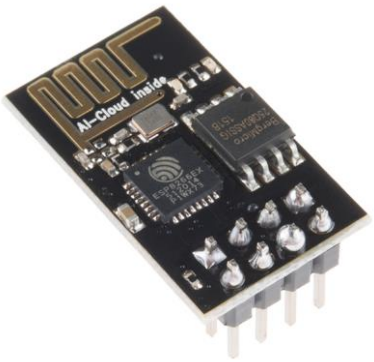

Fig. 12 – ESP8266

L’ ESP8266 ha la possibilità di essere programmato e funzionare in maniera autonoma, oppure può interfacciarsi con un microcontrollore e connetterlo ad internet.
Questo modulo ha una buona capacità di elaborazione e di archiviazione che permette, come detto in precedenza, di connettersi e gestire sensori e dispositivi specifici attraverso i suoi pin GPIO. [11]
L’ESP8266 ha le seguenti caratteristiche:

- 32-bit RISC CPU: Tensilica Xtensa LX106 con esecuzione a 80 MHz
- 64 KB di memoria RAM per le istruzioni e 96 KB di memoria RAM per i dati
- Supporta i protocolli WiFi IEEE 802.11 b/g/n
- Integrato interruttore TR, amplificatore di potenza e rete di adattamento
- Possibilità di connettersi a reti protette da password WEP o WPA/WPA2
- Abilitato alla comunicazione SPI e I^2C
- 8 pin di cui 2 per l’alimentazione, 2 GPIO, 2 per trasmissione dati e 2 per il controllo
- UART sui pin dedicati, oltre ad una trasmissione ad una sola UART che può essere attivata su GPIO2
- Possiede un ADC a 10 bit

Per un corretto funzionamento è necessario collegare a 3.3V i pin 3.3V, CH_PD E RST, rispettivamente alimentazione, chip power-down, e reset. [12]
I pin Tx e Rx vanno collegati rispettivamente ai pin 10, 11 i quali, grazie alla libreria SoftwareSerial sono diventati due pin di lettura e scrittura seriali.

3.7 Alimentazione

L’alimentazione è regolata dal circuito integrato LM2576 (Fig. 13), esso è un regolatore di tensione in grado di fornire in uscita una tensione costante di qualsiasi voltaggio, pilotando fino a 3A di corrente.
L’LM2576 richiede un numero minimo di componenti per il corretto funzionamento, è molto semplice da usare ed include una compensazione in frequenza.
Per un corretto funzionamento si è utilizzato il circuito descritto all’interno del datasheet e visibile nella figura sottostante, è possibile sia alimentare Arduino e tutti i sensori attraverso un

trasformatore che dia in uscita i 12/15V, oppure alimentandolo tramite delle batterie dotate di autoprotezione, in grado di ricaricarsi tramite pannello solare, fornendo ad Arduino sempre una tensione fissa regolabile attraverso un trimmer. [13]

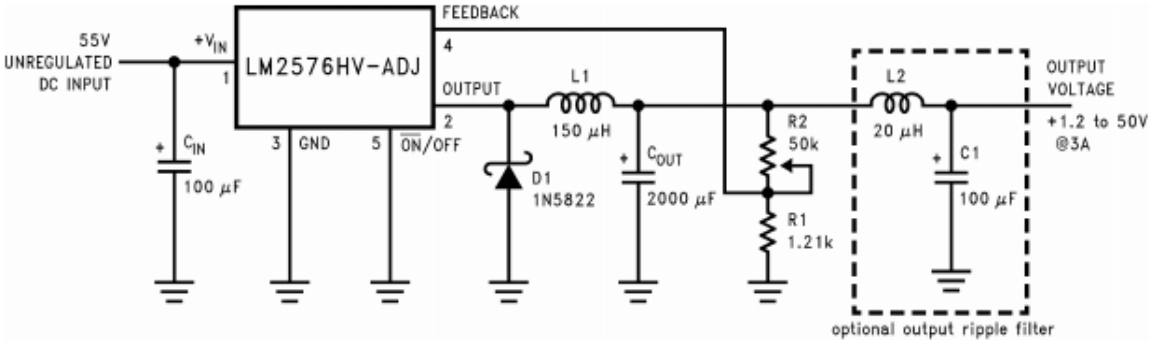

Fig. 13 – Schema circuito di alimentazione

Nel progetto si è modificata la resistenza del trimmer in modo che fornisse 7.5 V in uscita.

4. Software Utilizzato

In questo capitolo verrà trattato tutto il software utilizzato per la progettazione, partendo dall'ambiente di sviluppo di Arduino fino ad arrivare alle funzioni scritte per il corretto funzionamento del programma.

4.1 Ambiente di Sviluppo Arduino

L'ambiente di sviluppo per Arduino è scaricabile gratuitamente dal sito ufficiale. L'editor si presenta in veste molto minimalista, infatti si ha a disposizione solamente un editor di testo in cui scrivere il codice, un'area in cui saranno visualizzati gli eventuali errori a tempo di compilazione, la possibilità di aprire un monitor seriale con l'Arduino connesso via USB al computer, e poche altre finestre. Per scrivere un programma, che in gergo si chiama sketch, si utilizza l'editor di testo messo a disposizione dall'IDE. L'estensione di un file di codice Arduino è ".ino".

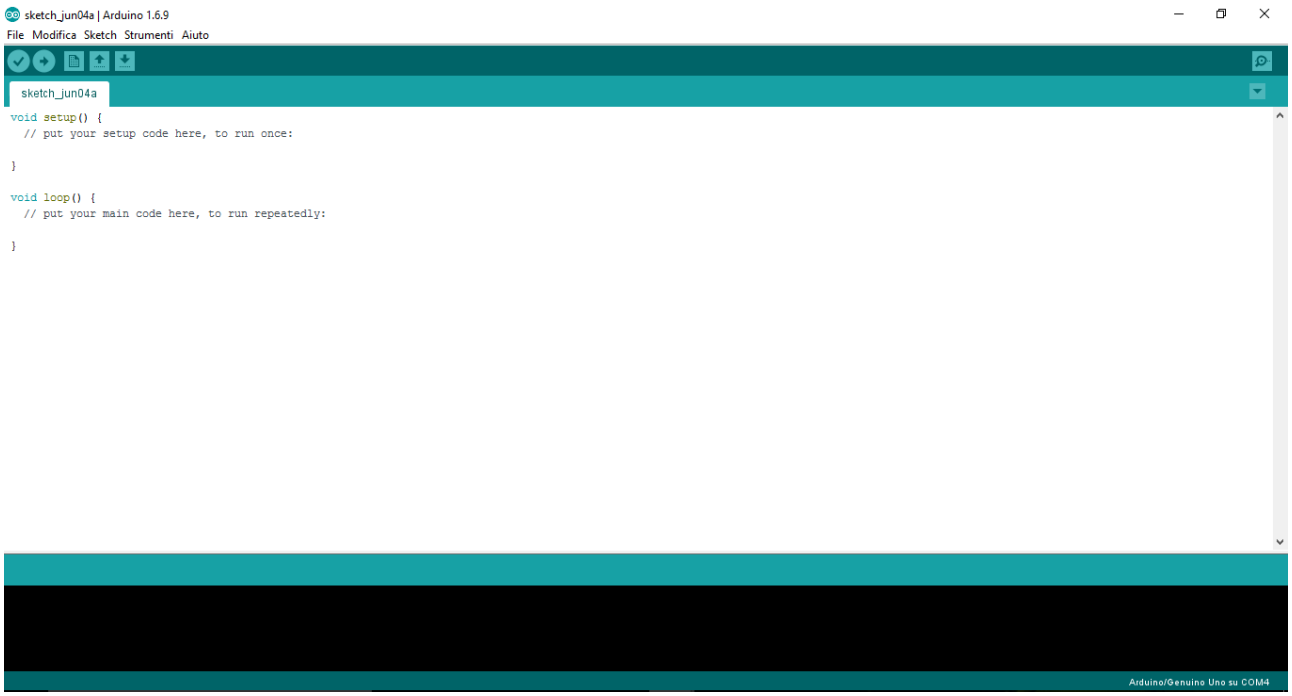

Fig. 14 - Ambiente di sviluppo Arduino

Gli unici controlli di testo che abbiamo a diposizione sono i classici copia/incolla, cerca/sostituisci. L'area dei messaggi sarà utile solo a tempo di compilazione in quanto se si commette un errore di

sintassi o semantica mentre si scrive il codice non saremo in nessun modo avvisati, fintanto che non viene compilato il codice. Arduino mette a disposizione una serie di librerie di base con le quali è possibile sviluppare dei programmi completi; sono inoltre disponibili alcuni esempi di progetto. Tutte le librerie ufficiali sono consultabili sul sito ufficiale di Arduino.
Se Arduino è correttamente collegato al pc, nella parte bassa dell'editor sarà notificato in quale porta USB è stato collegato. Tra le varie opzioni dell'editor è possibile importare le librerie di terze parti che risultano utili al fine di semplificare la programmazione e, come in molti altri linguaggi di programmazione, sarà necessario includere l'header di libreria all'interno del codice.

Ogni sketch di Arduino segue sempre le stesse regole, esso può essere diviso in due parti. Queste due parti sono i punti di ingresso di qualsiasi programma Arduino. Il primo punto è chiamato *setup*. È qui che vengono inizializzate tutte le risorse di Arduino, come ad esempio quali pin considerare ingressi o uscite, i servizi SD, Ethernet, WiFi ecc. Il secondo punto è chiamato *loop* e, come suggerisce il nome, Arduino ripeterà senza fine tutto il codice che sarà scritto all'interno di questo blocco, a meno di un evento esterno, come ad esempio un utente che preme il tasto reset. Come quasi tutti i linguaggi di questa tipologia, le zone esterne (prima e dopo) a queste due fasi possono essere utilizzate per l'inclusione di librerie, la definizione di variabili globali e di funzioni.

4.2 Librerie Utilizzate

Utilizzando vari sensori e componenti, scrivere un programma per gestirli tutti sarebbe stato lungo e difficoltoso; è per questo che sono state usate alcune librerie per snellire e rendere più semplice il software sia nella scrittura che nella comprensione durante la lettura.
Nel progetto sono state usate librerie ufficiali Arduino come:

- LiquidCristal
- Wire
- OneWire
- SoftwareSerial

Si è utilizzata inoltre una libreria non ufficiale creata dalla community Arduino chiamata DallasTemperature per il pilotaggio della sonda DS18B20.

4.2.1 LiquidCristal.h

Questa libreria permette ad una scheda Arduino di interfacciarsi ad uno schermo LCD con chipset Hitachi HD44780 (o compatibile), il quale si trova sulla maggior parte dei monitor LCD 16x2. La libreria lavora sia in modalità 4 o 8 bit.

4.2.2 SoftwareSerial.h

L'hardware Arduino è dotato di un supporto per la comunicazione seriale sui pin 0 e 1 (il quale comunica attraverso la stessa linea anche con il computer tramite la connessione USB). Questo hardware permette al chip Atmega di ricevere comunicazioni seriali anche mentre lavora su altri compiti, finché c'è spazio nel buffer seriale di 64 byte. La libreria SoftwareSerial è stata sviluppata per consentire la comunicazione seriale ad altri pin digitali di Arduino, utilizzando un software in grado di replicare le funzionalità dei pin 0 e 1 su altri pin, da qui il nome SoftwareSerial.
In questo modo è possibile avere più pin abilitati alla comunicazione seriale con velocità fino ai 115200 bps.

4.2.3 Wire.h

Questa libreria consente di comunicare con dispositivi I^2C e TWI. I pin dedicati alla trasmissione I^2C sono i pin A4 e A5. Il primo è la linea SDA (linea dati), mentre il secondo è SCL (linea di clock).
La libreria Wire utilizza 7 bit di indirizzo. Se si dispone di una scheda che utilizza l'indirizzo a 8 bit, si dovrà “far cadere” il bit più basso (cioè spostare il valore di un bit a destra), ottenendo un indirizzo compreso tra 0 e 127. Tuttavia gli indirizzi da 0 a 7 non vengono utilizzati perché sono riservati in modo che il primo indirizzo che può essere utilizzato sia 8.

4.2.4 OneWire.h

Questa libreria viene utilizzata per interfacciarsi con dispositivi OneWire, ossia quei dispositivi in cui la comunicazione con il microprocessore avviene su un solo cavo, come la sonda DS18B20 utilizzata nel progetto.
Per gestire la sonda oltre a questa libreria è stata utilizzata anche un'ulteriore libreria chiamata DallasTemperature.

4.2.5 DallasTemperature.h

Questa libreria è stata creata da parte della comunità Arduino ed è in grado di gestire parte dei sensori costruiti dalla Maxim tra cui la sonda DS18B20 utilizzata nel progetto. Questa libreria è nata inizialmente rielaborando la prima libreria OneWire ed in seguito, sono state aggiunte altre funzioni semplificando ulteriormente la modalità di interfacciamento con la sonda. [14]

4.3 Lettura della temperatura dalla sonda DS18B20

Come scritto in precedenza per interfacciarsi e leggere la sonda DS18B20 si è utilizzato la libreria DallasTemperature. Attraverso questa libreria è possibile comunicare con la sonda attraverso pochissime righe di codice.
Per prima cosa è necessario inizializzare le variabili per il corretto funzionamento.

```
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

E' quindi sufficiente, avvalendosi delle funzioni all'interno della libreria DallasTemperature, richiedere la temperatura e porla all'interno di una variabile, la quale sarà poi memorizzata nel sito internet.

```
sensors.requestTemperatures();
float temperature = sensors.getTempCByIndex(0);
```

4.4 Lettura della temperatura della sonda LM35

Il sensore LM35 è un sensore analogico che, come descritto in precedenza, dà in uscita una tensione proporzionale alla temperatura da esso rilevato.

Avendo un segnale analogico in ingresso per poterlo utilizzare è necessario trasformarlo in digitale, tramite l'ADC di Arduino.

Arduino è dotato di un convertitore A/D a 10bit (1024) e alla tensione di alimentazione di 5 volt corrisponde una risoluzione di 5/1024= 0.00488V (cioè circa 5mV); siccome l'LM35 fornisce in uscita 10mv per grado, si può dedurre che la massima precisione è di circa 0.5 gradi (5/10).

Per ottenere la temperatura in gradi Celsius si determinano dapprima i millivolt con la formula:

$$millivolt = (\frac{tempExtRead}{1024}) \times 5000$$

dove tempExtRead è il valore sul pin analogico dove è collegato il sensore, 5000 è il valore della tensione di alimentazione in millivolt e 1024 è il valore fornito dal convertitore A/D alla tensione di 5 volt.

In seguito si determinano i gradi Celsius con la formula:

$$GradiCelsius = millivolt/10$$

questo perché, come scritto in precedenza, l'LM35 fornisce 10mV ogni grado centigrado.

Per brevità si può utilizzare una sola formula che ci fornisce il risultato direttamente in gradi Celsius, come è stata utilizzata nel programma.

```
float tempExtRead = analogRead(tempPin);
float tempExt = (5.0 * tempExtRead * 100.0)/1024.0;
```

4.5 Lettura sensore HC-SR04

Come descritto in precedenza il sensore rileva il tempo che impiega l'onda da quando viene inviata a quando ritorna al sensore dopo che è stata riflessa da un corpo.

Si invia un impulso alto sul pin Trigger per almeno 10 microsecondi, a questo punto il sensore invierà il ping sonoro e aspetterà il ritorno delle onde riflesse, il sensore risponderà sul pin Echo con un impulso alto della durata corrispondente a quella di viaggio delle onde sonore, dopo 38 millisecondi si considera che non sia stato incontrato alcun ostacolo. Per sicurezza si aspettano in genere 50-60 millisecondi per far si che non vi siano interferenze con la misura successiva.

Per convertire l'intervallo di tempo misurato in una lunghezza, bisogna ricordare che la velocità del suono è di 331,5 m/s a 0 °C e di 343,4 m/s a 20 °C ed in generale varia secondo la relazione

$$v = 331{,}4 + 0{,}62 \times T$$

dove la temperatura T è misurata in °C.

Per comodità si è assunto di lavorare ad una temperatura ambiente di 20 °C e quindi la velocità del suono sarà di 343 m/s (approssimando) che vuol dire anche 0,0343 cm/microsecondi.

Quindi, ricordando che v=s/t (v: velocità, s: spazio, t: tempo) allora lo spazio percorso sarà:

$$s = 0{,}0343 \times t$$

però, per calcolare lo spazio percorso, bisogna tener conto che il suono percorre due volte la distanza da misurare (giunge sull'oggetto e ritorna indietro al sensore) quindi il valore di t ottenuto deve essere diviso per 2. La formula corretta per la misura dello spazio percorso è:

$$s = (0{,}0343 \times t)/2$$

Il tutto, per comodità, è racchiuso all'interno di una funzione denominata *readAltezza()*.

```
float readAltezza() {

//porta bassa l'uscita del trigger
digitalWrite( triggerPort, LOW );
//invia un impulso di 10microsec su trigger
digitalWrite( triggerPort, HIGH );
delayMicroseconds( 10 );
digitalWrite( triggerPort, LOW );

float durata = pulseIn( echoPort, HIGH );

float distanza = 0.034 * durata / 2;

return distanza;

delay(1000);
}
```

4.6 Trasferimento dati

4.6.1 Comandi AT

I comandi AT (o comandi Hayes) costituiscono un set di comandi (sotto forma di brevi stringhe di testo ASCII) per il controllo di dispositivi di comunicazione. Vennero introdotti dalla Hayes Communications nel 1977 come protocollo standard per controllare il loro Smartmodem; prima dell'introduzione di tale prodotto, infatti, i modem analogici venivano controllati mediante interfacce hardware dedicate che mal si prestavano a essere inserite nei microcomputer, che in quel periodo stavano conoscendo una forte diffusione. I comandi AT rivoluzionarono questo paradigma, permettendo il controllo di dispositivi quali modem attraverso una semplice linea seriale ed un set di comandi standard. Il set di comandi AT col passare del tempo divenne uno standard di fatto e venne utilizzato come sistema di interfacciamento per un gran numero di modem analogici. Il set prevede comandi per vari tipi di operazioni inerenti la linea telefonica e per il controllo dei registri interni del dispositivo connesso.
Il protocollo AT prevede una forma di comunicazione seriale di tipo comando-risposta. Il client invia un comando (terminato da un carattere di carriage return e opzionalmente da un new line) e riceve una risposta. [15]

I comandi AT (insieme alle loro funzionalità) utilizzati nel progetto sono raggruppati nella tabella seguente. (Fig. 15)

Comando AT	Risposta	Funzione
AT	OK	Controlla il corretto funzionamento
AT+RST	OK	Resetta il dispositivo
AT+CWLAP	AT+CWLAP +CWLAP:(4,"RochefortSurLac",-38,"70:62:b8:6f:6d:58",1) OK	Vengono visualizzati tutti gli access point disponibili
AT+CWJAP="SSID","Password"	OK	Connessione all'access point
AT+CIFSR	AT+CIFSR 192.168.0.105 OK	Prelevare indirizzo IP
AT+CWMODE=1 AT+CWMODE=2 AT+CWMODE=3	STA AP Both	Modalità di funzionamento Wifi
AT+ CIPMUX=0 AT+ CIPMUX=1	Single Multiple	Modalità di connessione (singola/multipla)
(CIPMUX=0) AT+CIPSTART =<type>,<addr>,<port> (CIPMUX=1) AT+CIPSTART=<id><type>,<addr>,<port>	id = 0-4, type = TCP/UDP, addr = IP address, port= port	Definizione del tipo connessione: TCP/UDP
AT+CIPCLOSE		Chiusa della connessione
(CIPMUX=0) AT+CIPSEND=<length> (CIPMUX=1) AT+CIPSEND= <id>,<length>		Invio dati

Fig. 15 – Tabella comandi AT

4.6.2 Connessione Wifi tramite ESP8266

La connessione al modem avviene utilizzando il dispositivo ESP8266, il quale, attraverso i comandi AT è in grado di interfacciarsi ad un qualsiasi router ed eventualmente inviare dati ad un determinato server.

Anche in questo caso si è optato per racchiudere tutte le istruzioni all'interno di una funzione booleana chiamata *connectWiFi()*, che risulta vera in caso di corretta connessione.

```
boolean connectWiFi(){
  Serial.println("AT+CWMODE=1");
  softser.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  softser.println(cmd);
  delay(5000);
  if(softser.find("OK")){
    lcd.clear();
    lcd.print("Connesso");
    Serial.println("Connesso");
    return true;
  }else{
    return false;
  }
}
```

Attraverso determinati comandi AT ci si può connettere a qualsiasi router protetto semplicemente da una password WAP/WEP.

E' sufficiente impostare la connessione di tipo STA attraverso l'apposito comando AT+CWMODE =1. In seguito è sufficiente, sempre con un comando AT (AT+CWJAP) connettersi all'access point desiderato inserendo al posto di SSID il nome del router e al posto di PASS la password.

4.7 ThingSpeak

ThingSpeak è una piattaforma applicativa appositamente sviluppata per semplificare lo sviluppo di dispositivi IoT (Internet of Thing).
ThingSpeak permette di realizzare un'applicazione tramite i dati raccolti da uno o più sensori dislocati in posizioni differenti, anche molto lontani tra di loro. [16]

Le principali caratteristiche che lo contraddistinguono sono:

- raccolta dati in real-time;
- elaborazione dati;
- visualizzazione dei dati;
- applicazioni e plug-in;
- possibilità di scarica i dati e leggerli su un foglio di calcolo

Alla base di ThingSpeak ci sono i cosiddetti canali; un canale è dove i dati vengono inviati e memorizzati.
Ogni canale ha la possibilità di creare fino 8 campi per la memorizzazione di 8 dati provenienti da altrettanti sensori; inoltre sono presenti 3 ulteriori campi per la geolocalizzazione.

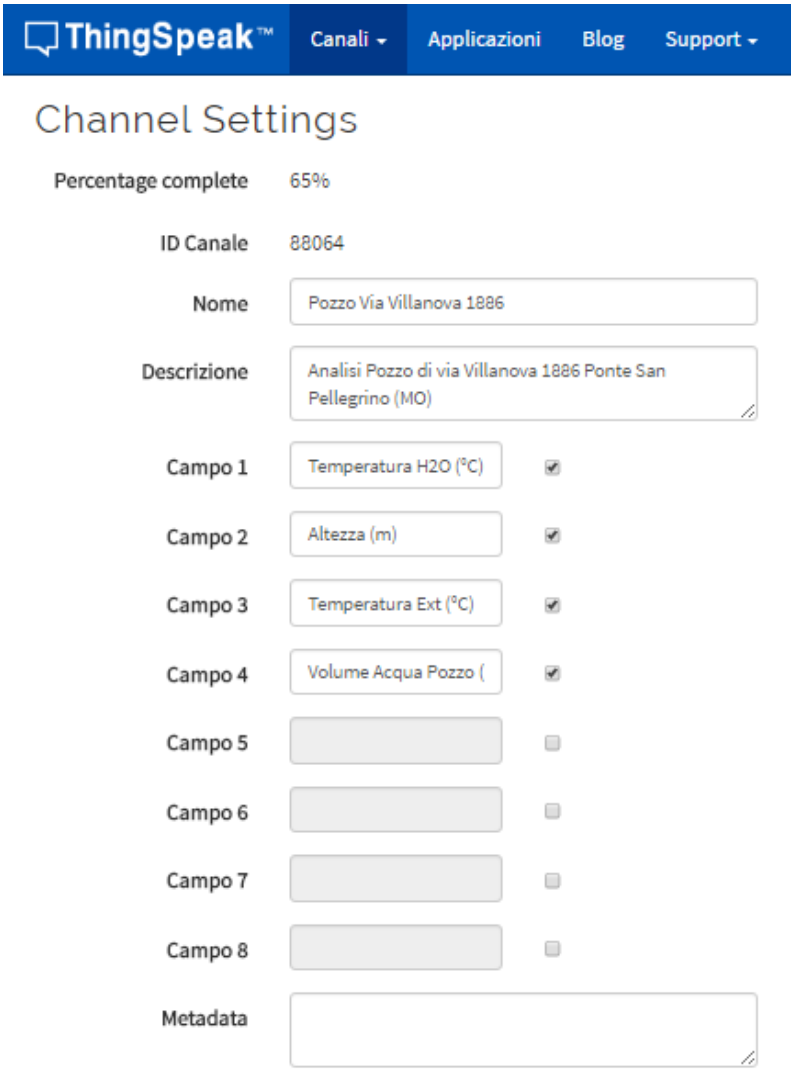

Fig. 16 – Pannello di comando ThingSpeak

Per poter utilizzare tutte le funzionalità di ThingSpeak è sufficiente registrarsi al sito internet in maniera gratuita e creare il proprio canale con i relativi 8 campi.
Ogni canale avrà la sua personale chiave API che permetterà l'aggiornamento dei dati.

4.7.1 Caricamento dati su ThingSpeak

Per poter caricare è necessario trasformarli da numeri con virgola (Float) a "lettere" (String).
Per fare ciò si è utilizzata la funzione dtostrf() la quale è contenuta nelle librerie interne di Arduino.

Oltre alla funzione è necessario creare una variale di buffer in cui il dato verrà salvato temporaneamente prima di essere salvato definitivamente sulla variabile in formato String che verrà inviata online.

```
String temp = dtostrf(temperature, 4, 1, buffer);
```

Dopo aver trasformato tutti i valori in stringhe, attraverso la funzione *updateData()* avviene il caricamento dei dati.

```
void updateData(String tempF, String altezF, String tempExtF, String volumeStr){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "api.thingspeak.com";
  cmd += "\",80";
  Serial.println(cmd);
  softser.println(cmd);
  delay(2000);
  if(softser.find("Error")){
    return;
  }

 cmd = GET;
 cmd += "&field1=";
 cmd += tempF;
 cmd += "&field2=";
 cmd += altezF;
 cmd += "&field3=";
 cmd += tempExtF;
 cmd += "&field4=";
 cmd += volumeStr;
 cmd += "\r\n";
 Serial.print("AT+CIPSEND=");
 softser.print("AT+CIPSEND=");
 Serial.println(cmd.length());
 softser.println(cmd.length());
 delay(2000);

if(softser.find(">")){
  Serial.println(cmd);
  softser.print(cmd);

}else{
  softser.println("AT+CIPCLOSE");
  Serial.println("AT+CIPCLOSE");
  contatoreNoScritture++;

}}
```

Analogamente alla connessione Wifi si utilizzano i comandi AT, in particolare il comando AT+CIPSTART=" "; per inviare i dati, la relativa chiave API e il comando AT+CIPSEND per verificare che la trasmissione sia avvenuta con successo.

4.8 Accorgimenti Software

Il progetto con le funzioni sopracitate funziona in maniera corretta, sono però stati fatti accorgimenti per un funzionamento migliore ed in grado di resettarsi in caso di errore.

4.8.1 Temporizzazione del progetto

La lettura dei sensori ed il conseguente aggiornamento dei dati è temporizzato.
La temporizzazione avviene attraverso la funzione *delay()* la quale mette in pausa Arduino finchè non è trascorso il tempo scritto all'interno della funzione.
Il resto della temporizzazione avviene grazie alla funzione *millis()* la quale salva all'interno di una variabile unsigned long il tempo trascorso dall'inizio dell'esecuzione del programma in millisecondi.
Salvando il tempo trascorso al termine di un determinato processo è possibile confrontarlo con una variabile fissa corrispondente ai millisecondi di un periodo di tempo fissato.

Nel progetto si sono utilizzati 3 tempi prefissati:

- 10 minuti = 600000 millisecondi;
- 1 ora = 3600000 millisecondi;
- 1 giorno = 86400000 millisecondi;

Ogni 10 minuti è il tempo in cui vengono letti i dati e aggiornati su ThingSpeak.
Si è scelto questo tempo perché la temperatura dell'acqua non varia in maniera veloce, calcolando anche il volume di acqua preso in considerazione.

L’ ora viene utilizzata come tempo di controllo per il corretto invio dei dati, nel caso non vengano scritti i dati per più di un’ora (quindi alla sesta scrittura mancata) viene resettato il dispositivo e viene inviato un Tweet.

La variabile temporale che indica il giorno serve per resettare il conteggio dei Tweet scritti in quanto non è possibile scrivere lo stesso Tweet prima che siano trascorse 24 ore.

4.8.2 Allarme Tweet e reset del dispositivo

Per automatizzare il dispositivo si è pensato di creare un account Twitter per il dispositivo sul quale, mediante le chiavi API di ThingSpeak, verranno inviati dei Tweet quando la temperatura dell’acqua del pozzo supera una determinata soglia, facendo in modo però che non vengano rimandati allarmi prima che sia trascorsa un’ora.

```
if((temperature >30)&&(currentMillis - previousMillis>intervalHour)){
  previousMillis = millis();
  contTweet++;
  allarmeTweet(temp);
}
```

Attraverso un semplice if che controlla simultaneamente temperatura e tempo trascorso, va ad una funzione *allarmeTweet()* la quale attraverso il sito ThingSpeak e delle chiavi API, in maniera analoga all’update dei dati, viene scritto un tweet con scritto: “Attenzione, temperatura a “temp” °C”.

Nel caso il dispositivo perda la connessione wifi è stato previsto un reset software il quale, in seguito a sei mancate scritture dei dati resetterà il dispositivo in modo che sia possibile per lui ristabilire la connessione.
In seguito al reset verrà inviato un tweet con scritto che non era presente una connessione Wifi e che il dispositivo è stato resettato attraverso la funzione *errorescritturaTweet().*

5. Risultati e Sviluppi futuri

Tutti i dispositivi hardware utilizzati sono stati posizionati all'interno di una scatola di derivazione impermeabile.

A questa scatola sono stati praticati dei fori e delle aperture nella parte frontale per il display, per il pulsante di accensione della retroilluminazione dello schermo Lcd, per l'interruttore di accensione e di spegnimento del dispositivo.

Nella parte inferiore è stato predisposto una presa per la ricarica delle batterie che può avvenire sia attraverso un caricabatterie sia attraverso un pannello solare.

Nella parte superiore attraverso un passacavo è stato predisposto un ingresso per i cavi dei sensori che devono essere posizionati sul pozzo e quindi al di fuori della scatola.

Fig.17 – Prototipo posizionato sul pozzo

Dopo aver mantenuto in funzione il dispositivo, è possibile visualizzare, sul corrispondente link (https://thingspeak.com/channels/88064), l'andamento di tutte e tre le variabili misurate (più il

valore calcolato del volume), le quali sono state salvate e graficate su ThingSpeak con sull'asse delle ascisse il tempo trascorso (portata di 500 letture), mentre sull'asse delle ordinate il valore registrato.

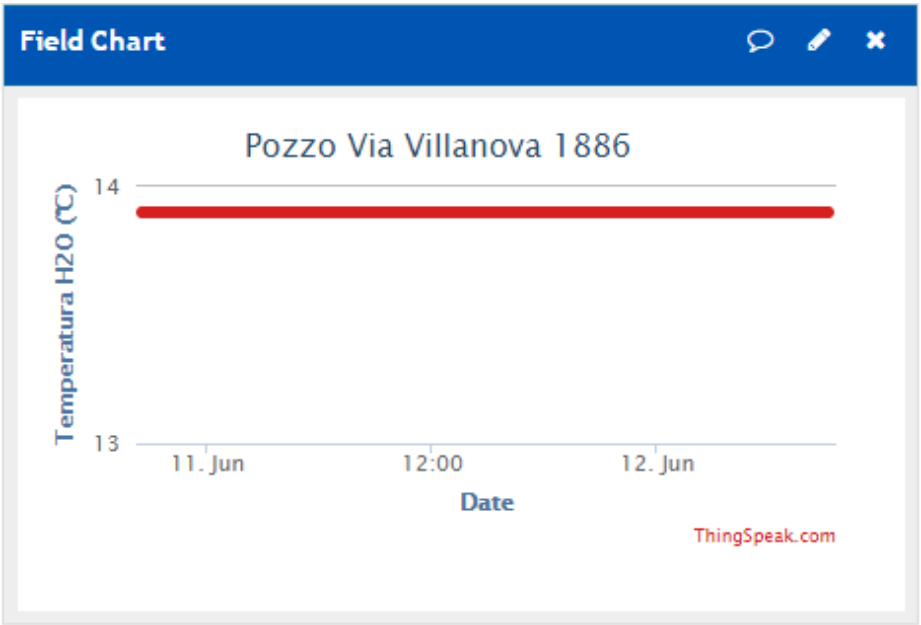

Fig. 18.a – Grafico temperatura acqua

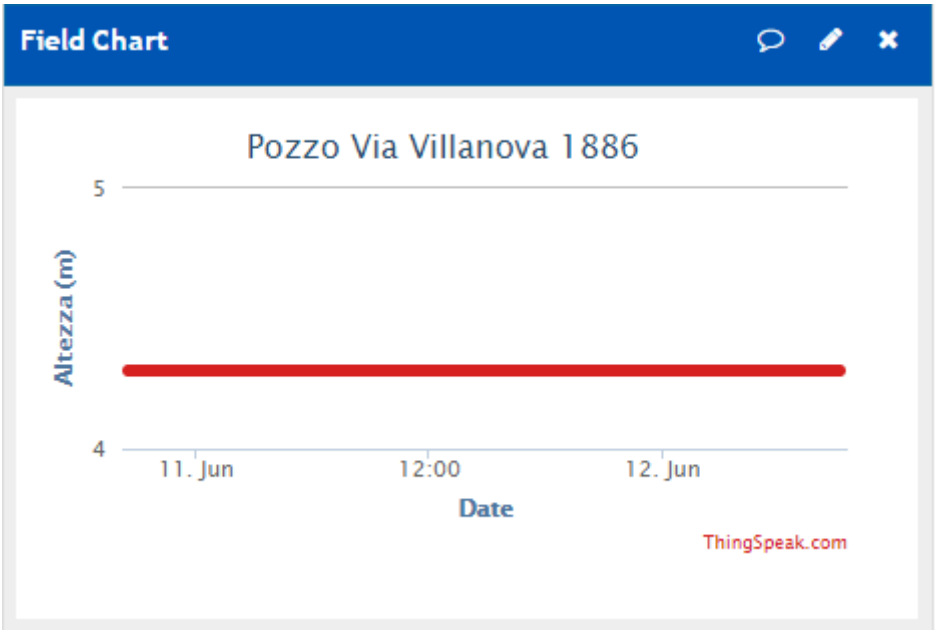

Fig. 18.b – Grafico altezza acqua

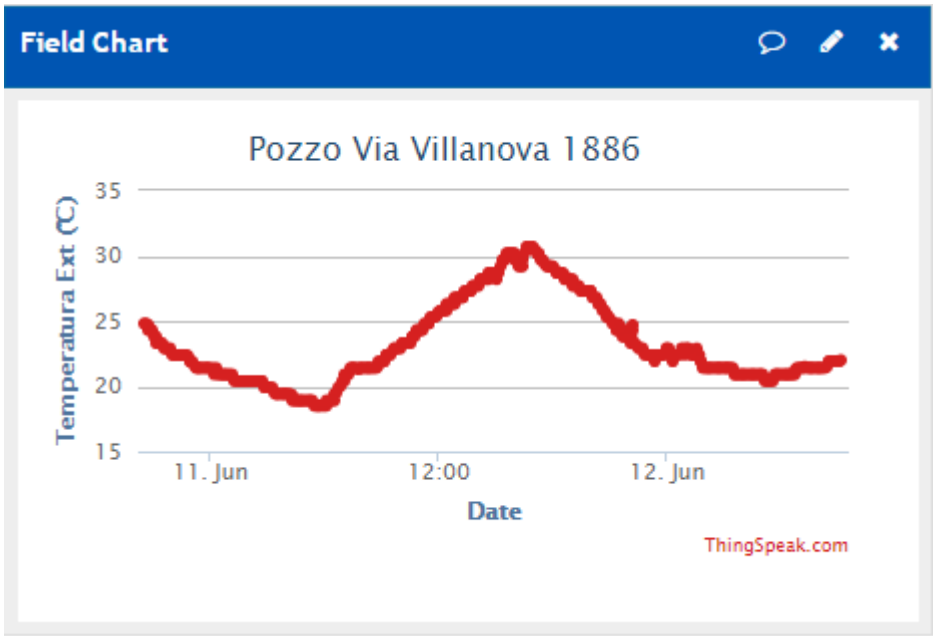

Fig. 18.c – Grafico temperatura esterna

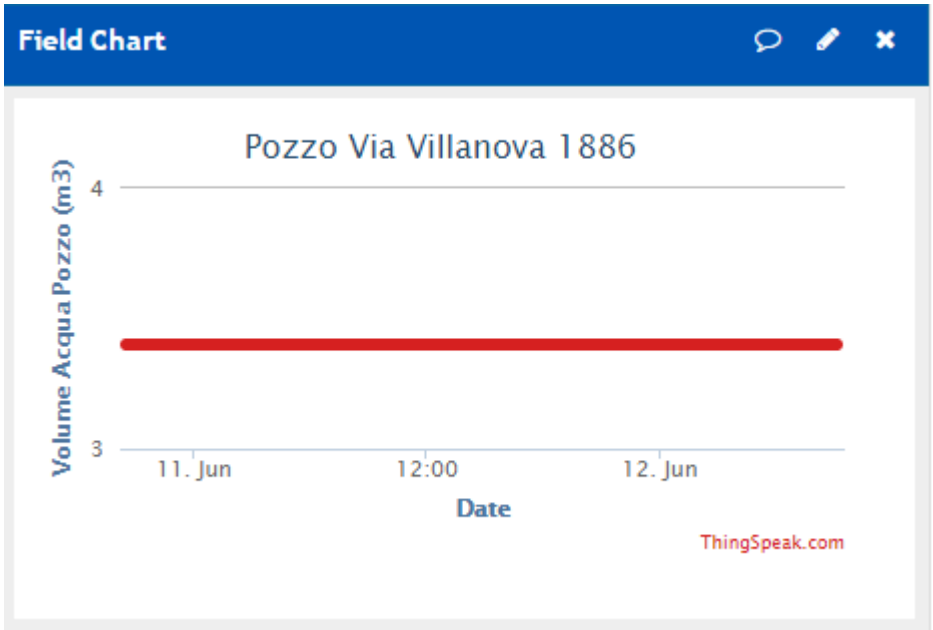

Fig. 18.d – Grafico volume acqua

Si è notato che la temperatura dell'acqua non varia in maniera apprezzabile rimanendo stabile alla temperatura di circa 14°C.

Anche l'altezza dell'acqua, rimane stabile all'altezza di 4,1m; quest'ultima però, quando si accende il motorino per prelevare l'acqua del pozzo cala circa di 10 cm ogni 10 minuti di utilizzo.
Pian piano l'acqua ritornerà a salire sfruttando la falda sottostante impiegando però molto più tempo, risalendo con una media di 10 cm ogni 40 minuti circa.

E' stata predisposta una porta all'alimentazione per permettere il caricamento delle batterie tramite pannello solare il quale sarà installato il prima possibile, al momento questa porta viene utilizzata per ricaricare le batterie attraverso un alimentatore che fornisce 15V e 0.8mA.

Bibliografia

[1] – Terremoti

- Costruire in Zona Sismica: Guida tecnica ai particolari costruttivi, strutturali e normativi
  – Nazzareno Centini

[2] – Articoli fenomenologia secondaria durante i terremoti

- http://ambiente.regione.emilia-romagna.it/geologia/temi/geologia/fenomeni-geologici-inusuali-in-emilia-romagna

[3] – Articolo riscaldamento acqua di un pozzo

- http://gazzettadimodena.gelocal.it/modena/cronaca/2015/10/22/news/acqua-calda-nel-pozzo-di-medolla-oggi-prelievi-della-regione-1.12310302?

[4] – Mircoprocessore Atmel,

- http://www.atmel.com/Images/doc8161.pdf

[5] – Sito Ufficiale Arduino, Arduino Uno

- http://arduino.cc/en/Main/arduinoBoardUno

[6] – Datasheet HC-SR04

- http://www.micropik.com/PDF/HCSR04.pdf

[7] – Datasheet DS18B20

- https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[8] – Funzionamento sensori

- Modern Sensor Handbook - Pavel Ripka Alois Tipek

[9] – Datasheet Display Lcd

- https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[10] – Datasheet LM35

- www.ti.com/lit/ds/symlink/lm35.pdf

[11], [12] – Specifiche Esp8266

- https://www.adafruit.com/datasheets/ESP8266_Specifications_English.pdf
- http://wiki.piffa.net/index.php/ESP8266

[13] – Datasheet LM2576

- http://www.ti.com/lit/ds/symlink/lm2576.pdf

[14] – Libreria Dallas Temperature

- https://milesburton.com/Dallas_Temperature_Control_Library#Benefits

[15] – Comandi AT

- Futura Elettronica n°

[16] – ThingSpeak

- https://thingspeak.com/