5) PROGETTO PRATICO

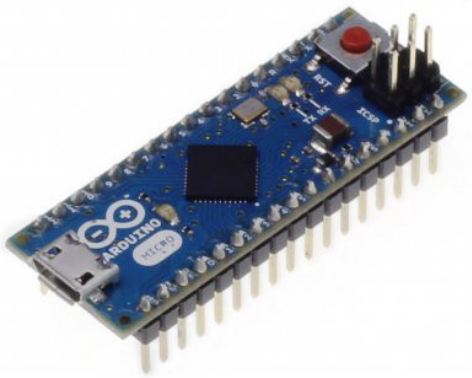
5.1) FINALITÁ DEL PROGETTO

Nell'ottica di realizzare un progetto che permettesse di ottenere un risparmio energetico, ho pensato ad un campo indispensabile per la sopravvivenza di tutta l'umanità, ma che porta anche un grosso consumo di energia e in particolare di acqua, ovvero l'agricoltura.

Sta infatti diventando sempre più importante per tutti ridurre gli sprechi idrici in qualsiasi campo e ridurre e riuscire a eliminarli in questo campo porterebbe a un risparmio non trascurabile, sia economico che energetico.

Quindi ho ideato un sistema di irrigazione automatica che attraverso una sonda nel terreno misuri l'umidità di questo e quindi analizzi questo dato: nel caso esso sia inferiore alla soglia scelta dal cliente e l'orario sia compreso in una fascia oraria, anche questa impostabile dall'utilizzatore, il sistema provvederà ad irrigare attraverso una elettrovalvola che svolgerà il ruolo di rubinetto.

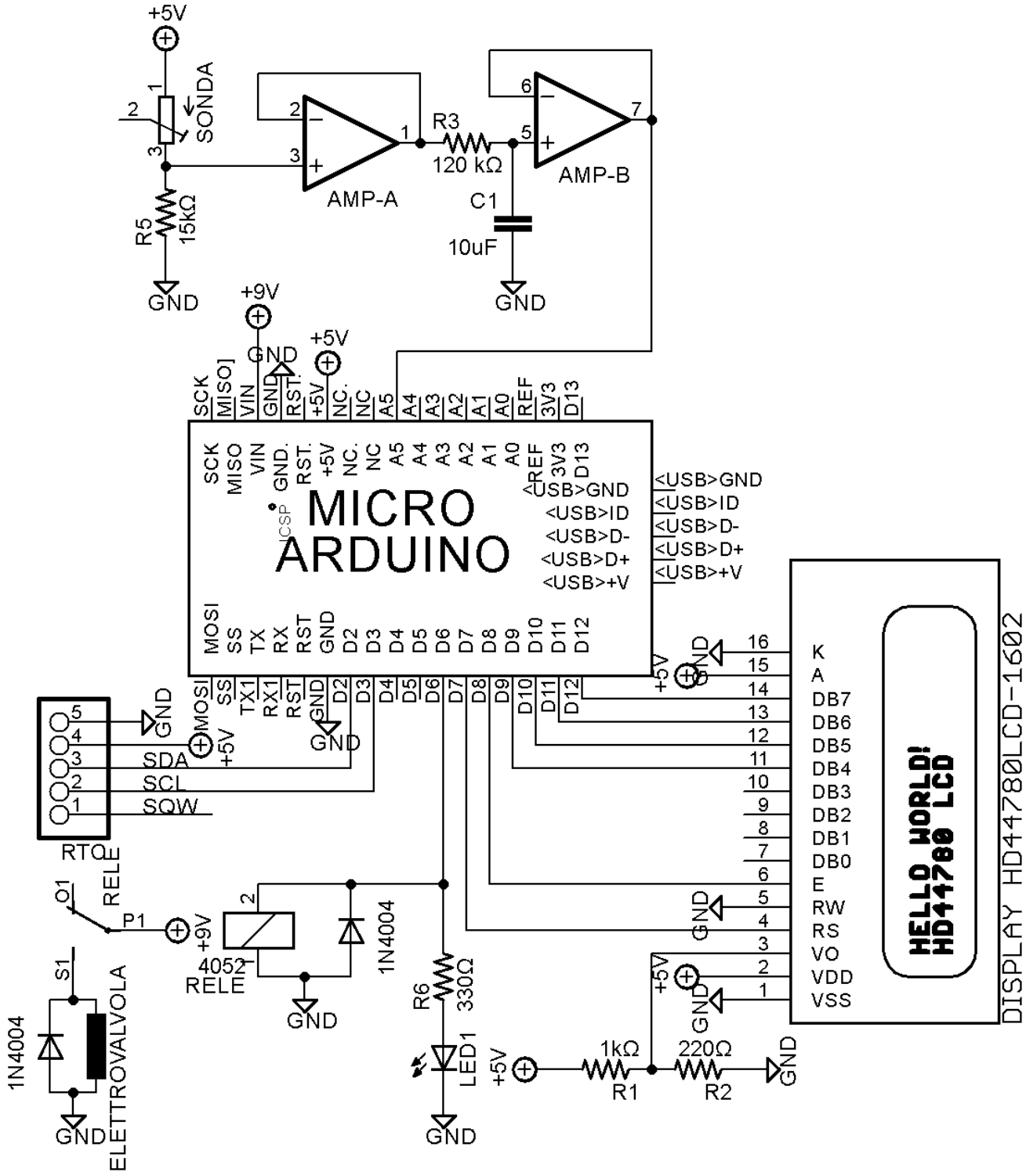
5.2) STRUTTURA



Per realizzare il sistema ho subito deciso di utilizzare la scheda μC Arduino, ed in particolare un Arduino Micro, grazie alla sua compattezza. Esso ha 20 pin digitali di input/output (di cui 7 possono essere usati come uscita analogica (ottenuta tramite il PWM) e 12 possono essere usati come ingresso analogico) e un collegamento micro-USB integrato. Esso include inoltre un chip di alimentazione che può ricevere tensioni in ingresso da 6 a 12 V.
Per aumentare ancora il risparmio energetico ho implementato la modalità sleep (un 'sonno profondo', con il processore centrale Atmega32u4 e tutte le periferiche spente, che porta la scheda a consumare pochi μA) fra una rilevazione e l'altra.
Per avere sempre l'orario e la data corretti collego all'Arduino un modulo Real Time Clock (in grado di mantenere l'orario in registro anche durante la modalità sleep e in assenza di alimentazione grazie alla batteria interna) con chip DS1307 e collegamento I^2C.
Includerò al circuito un display LCD con chip HD44780 per visualizzare l'orario, i dati rilevati e informazioni varie.

In ingresso avrò come trasduttore di umidità una sonda composta da due elettrodi che dovrò conficcare nel terreno. Questa mi permetterà di misurare l'umidità nel terreno sotto forma di una resistenza variabile, in quanto il terreno fra gli elettrodi si comporterà come un carico resistivo inversamente proporzionale al valore dell'umidità. Come circuito di condizionamento ho quindi utilizzato un partitore di tensione composto dalla sonda e da una resistenza da 15 kΩ.
In uscita invece dovrò pilotare una elettrovalvola, ovvero un rubinetto la cui apertura e chiusura dipende dalla presenza o no di corrente sul suo avvolgimento interno. Questa corrente, tuttavia, è troppo alta per l'uscita dei pin digitali di Arduino, al massimo 40mA. Dovremo quindi realizzare un circuito di pilotaggio composto da un relè che provvederà a collegare l'elettrovalvola all'alimentazione quando imposto da Arduino.

5.3) PROGETTO



L'elettrovalvola va alimentata a 9V, per cui userò una alimentazione da 9V che sfrutterò anche per la scheda Arduino e, attraverso il suo chip di alimentazione, tutti gli altri componenti, collegati all'uscita 5V della scheda.

Il circuito di condizionamento per la sonda a partitore di tensione ha bisogno di un adattamento di impedenza in uscita per evitare disturbi, per cui pongo in cascata un inseguitore di tensione. Metterò

quindi un filtro passa-basso per eliminare l'eventuale rumore, e infine metto un altro inseguitore per adattare l'impedenza d'uscita del filtro.
Dimensiono il filtro passa-basso: il segnale che ho in entrata non varia molto velocemente, per cui la frequenza di taglio sarà abbastanza bassa, più precisamente:

$$f_T = \frac{1}{t_{CAMP}} = \frac{1}{8\,s} = 125\,mHz$$

Scelgo un condensatore da 10 μF, quindi trovo il valore appropriato della resistenza:

$$R = \frac{1}{2\pi f_t C} = \frac{1}{2*\pi*\left(125*10^{-3}\right)*10^{-5}} = \frac{1}{\pi*250*10^{-8}} = 127\,k\,\Omega$$

La resistenza in commercio con valore più vicino a questo valore è 120 kΩ, per cui sceglierò questa.

Inserirò un LED sul canale in uscita verso il relè di pilotaggio della elettrovalvola per monitorare l'apertura di quest'ultima. Userò un LED con questi parametri:

$$I_{LED} = 10\,mA$$
$$V_{LED} = 2\,V$$

Ora dovrò dimensionare la resistenza da applicare al LED:

$$R_{LED} = \frac{V_{CC} - V_{LED}}{I_{LED}} = \frac{3}{10^{-2}} = 300\,\Omega$$

La resistenza in commercio con valore più vicino a questo valore è 330 Ω, per cui sceglierò questa.

Sia il relè utilizzato che la elettrovalvola contengono carichi induttivi che creano correnti di ritorno, per cui porrò un diodo in configurazione inversa su entrambi.

5.4) LA PROGRAMMAZIONE

Il programma fa si che che l'Arduino fra una lettura e l'altra (che include il controllo della necessità di irrigare) e la successiva vada in modalità sleep per 8 secondi (il tempo massimo in cui può stare

in sleep, dopo il quale viene risvegliato dal timer WatchDog). Per questo ho utilizzato la libreria "LowPower".
Per interfacciarmi con il modulo Real Time Clock ho usato le librerie "Wire" e "RTClib".

Ho realizzato il software con in mente l'idea di creare un programma che desse la possibilità di comunicare via seriale con un qualsiasi PC via collegamento USB (per permettere di settare orario, soglia di umidità e orari di irrigazione ma anche per effettuare il debug o semplicemente per leggere i dati letti dal Arduino sullo schermo del PC).
Tramite un monitor seriale sul PC si possono inviare i seguenti comandi:

- q = Resetta il tempo al tempo nel momento in cui abbiamo caricato il programma dal PC
- w (seguito da un numero da 1 a 100) = Reimposta la soglia di umidità al valore inviato
- e = Abilita/Disabilita la modalità risparmio energia
- r (seguito da un numero da 0 a 23) = Reimposta l'ora iniziale della fascia di tempo in cui irrigare all'ora inviata
- t (seguito da un numero da 0 a 23) = Reimposta l'ora finale della fascia di tempo in cui irrigare all'ora inviata

La comunicazione seriale non è tuttavia possibile utilizzando lo sleep nel programma (in quanto questo fa terminare il collegamento seriale, mandando in crash il monitor seriale sul PC). Per questa ragione ho creato una variabile booleana chiamata 'rispEn' che deve essere posta alla condizione *true* perché l'Arduino attivi la sleep mode, altrimenti la sostituisce con un semplice delay. La variabile parte all'avvio (o al reset) come *false* in modo da fare avviare la scheda con la modalità sleep disabilitata per permettere di modificare le variabili desiderate all'inizio, quindi per attivare la modalità sleep sarà sufficiente inviare il carattere "e" via seriale.

Questo è il software completo:

```
/**  Richiamo le librerie necessarie  **/
#include <Wire.h>  //Includo la libreria per il collegamento seriale I2C (necessario per il modulo Real Time Clock)
#include "RTClib.h"  //Liberia per gestire il Real Time Clock
#include <LiquidCrystal.h>  //Libreria per gestire lo schermo LCD
#include "LowPower.h"  //Libreria per gestire la modalità sleep (risparmio energia)

/**  Creo le variabili di funzionamento  **/
#define sensoreUmidita A5  //Pin del sensore di umidità del terreno
#define elettrovalvola 6  //Pin di collegamento verso la valvola
LiquidCrystal lcd(7,8,9,10,11,12);  //Pin di collegamento del LCD
RTC_DS1307 RTC;

/**  Creo le variabili globali per il programma  **/
int giorno, mese, anno, ore, minuti, secondi;
int umidita=0, primaRiga, sogliaUmidita=50, oraInizio, oraFine;
```

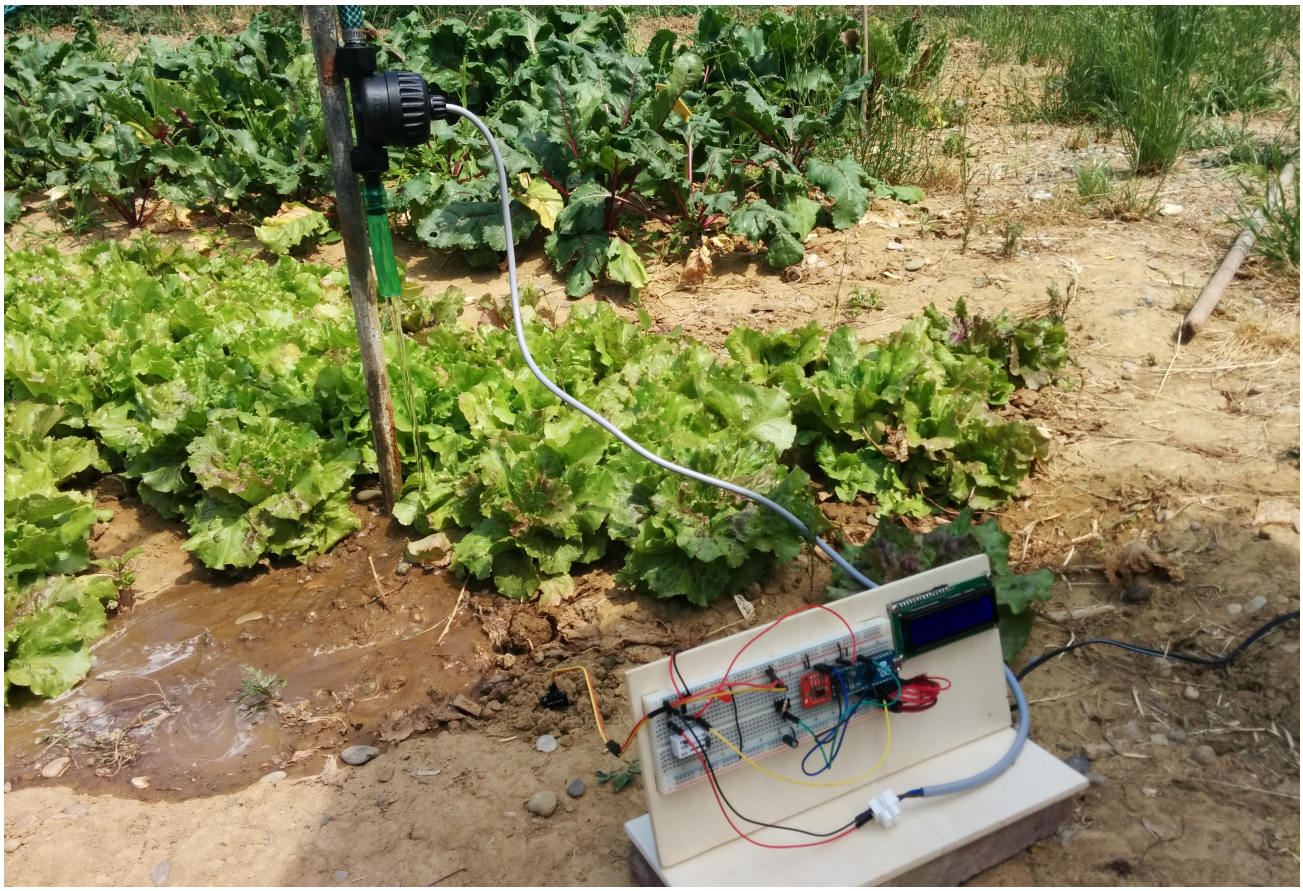
```
boolean rispEn=false, fasciaOraria;

void setup () {
  /**  Avvio la seriale, il collegamento I2C con il modulo Real Time Clock e l'LCD  **/
  Serial.begin(9600);
  Wire.begin();
  RTC.begin();
  controlloRTC();
  RTC.sqw(0);
  lcd.begin(16,2);
  pinMode(elettrovalvola,OUTPUT);

  tempo();
  if(4<=mese<=9){ //In primavera e estate dovrà innaffiare di notte
    oraInizio=0;
    oraFine=6;
  }
  else{  //In autunno e inverno dovrà innaffiare di pomeriggio
    oraInizio=12;
    oraFine=18;
  }

  stampaLCD();
  lcd.setCursor(0,1);
  lcd.print("CALIBRAZIONE...");
  dormi();
}

void loop () {
  controlloRTC();
  if(! rispEn) controlloSeriale();
  tempo();
  stampaLCD();
  umidita=map(analogRead(sensoreUmidita),0,1023,0,100);
  if(! rispEn) stampaSeriale();
  lcd.setCursor(0,1);
  if(umidita<sogliaUmidita && fasciaOraria){
    digitalWrite(elettrovalvola,HIGH);
    lcd.print("IRRIGANDO (");
    lcd.print(umidita);
    lcd.print("%)  ");
  }
  else{
    digitalWrite(elettrovalvola,LOW);
    lcd.print("UMIDITA': ");
    lcd.print(umidita);
    lcd.print("%   ");
  }
  dormi();
}

/**  Creo la funzione che controlla se il modulo Real Time Clock funziona correttamente  **/
void controlloRTC(){
  while (! RTC.isrunning()) {
    lcd.setCursor(0,0);
    lcd.print("RTC NON FUNZIONA");
    delay(2000);
    if(! rispEn) controlloSeriale();
  }
}
```

```
/**  Creo la funzione che controlla se è arrivato qualcosa via Seriale; in caso affermativo, esegue l'azione corrispondente al comando inviato  **/
void controlloSeriale(){
  if(Serial.available()>0){
    int y=Serial.read(), buf=Serial.available(), a[buf];
    for(int x=0; x<buf; x++) a[x]=Serial.read();
    switch(y){
      /* COMANDI INVIABILI VIA SERIALE:
       * q = Resetta il tempo al tempo nel momento in cui abbiamo caricato il programma dal PC l'ultima volta
       * w (seguito da un numero da 1 a 100) = Reimposta la percentuale di umidità sotto cui si apre la valvola al valore inviato
       * e = Abilita/Disabilita la modalità risparmio energia
       * r (seguito da un numero da 0 a 23) = Reimposta l'ora di inizio della fascia di tempo in cui irrigare
       * t (seguito da un numero da 0 a 23) = Reimposta l'ora finale della fascia di tempo in cui irrigare
       */
      case(113):  //113 ASCII = q
      RTC.adjust(DateTime(__DATE__, __TIME__));
      Serial.println("ORARIO REIMPOSTATO");
      break;

      case(119):  //119 ASCII = w
      if(buf==1) sogliaUmidita=a[0]-48;  //48 ASCII = 0
      if(buf==2) sogliaUmidita=(a[0]-48)*10 + a[1]-48;
      if(buf>=3) sogliaUmidita=100;
      Serial.print("NUOVA SOGLIA UMIDITA': ");
      Serial.print(sogliaUmidita);
      Serial.println("%");
      break;

      case(101):  //101 ASCII = e
      rispEn=!rispEn;
      if(rispEn){
        Serial.println("SLEEP ABILITATO");
        delay(100);
        Serial.end();
      }
      else Serial.println("SLEEP DISABILITATO");
      break;

      case(114): //114 ASCII = r
      if(buf==0) Serial.print("ERRORE, DEVI INSERIRE UN VALORE DA 0 A 23");
      if(buf==1) oraInizio=a[0]-48;
      if(buf>=2) oraInizio=(a[0]-48)*10 + a[1]-48;
      Serial.println("ORA DI INIZIO IRRIGAZIONE REIMPOSTATA");
      break;

      case(116):  //116 ASCII = t
      if(buf==0) Serial.print("ERRORE, DEVI INSERIRE UN VALORE DA 0 A 23");
      if(buf==1) oraFine=a[0]-48;
      if(buf>=2) oraFine=(a[0]-48)*10 + a[1]-48;
      Serial.println("ORA DI FINE IRRIGAZIONE REIMPOSTATA");
      break;

    default:
      Serial.print("COMANDO SCONOSCIUTO (ASCII ");
      Serial.print(y);
      Serial.println(")");
    }
  }
```

```
}

/**  Creo la funzione che stampa via seriale data e ora ottenute con la funzione 'tempo' e umidità  **/
void stampaSeriale(){
  Serial.print(giorno);
  Serial.print("/");
  Serial.print(mese);
  Serial.print("/");
  Serial.print(anno);
  Serial.print(" - ");
  Serial.print(ore);
  Serial.print(":");
  Serial.print(minuti);
  Serial.print(":");
  Serial.print(secondi);
  Serial.print(" - ");
  Serial.print(umidita);
  Serial.print("% [Ore ");
  Serial.print(oraInizio);
  Serial.print("-");
  Serial.print(oraFine);
  Serial.print(", Soglia ");
  Serial.print(sogliaUmidita);
  Serial.println("%]");
}

/**  Creo la funzione che ricava data e ora dal modulo Real Time Clock  **/
void tempo(){
  DateTime now = RTC.now();
  giorno=now.day();
  mese=now.month();
  anno=now.year();
  ore=now.hour();
  minuti=now.minute();
  secondi=now.second();
     if(oraInizio==oraFine || (oraInizio<oraFine && ore>=oraInizio && ore<=oraFine) || (oraInizio>oraFine &&
(ore>=oraInizio || ore<=oraFine))) fasciaOraria=true;
  else fasciaOraria=false;
}

/** Creo la funzione che stampa la prima riga sul display LCD  **/
void stampaLCD(){
  lcd.setCursor(0,0);
  //Alterneremo l'orario attuale, l'orario di irrigazione e soglia di umidità
  if(primaRiga<3) primaRiga++;
  else primaRiga=1;
  if(primaRiga==1){
    lcd.print(giorno);
    lcd.print("/");
    lcd.print(mese);
    lcd.print("/");
    lcd.print(anno);
    lcd.print("  ");
    if(ore<10) lcd.print("0");
    lcd.print(ore);
    lcd.print(":");
    if(minuti<10) lcd.print("0");
    lcd.print(minuti);
  }
  if(primaRiga==2){
```

```
    lcd.print("SOGLIA: ");
    lcd.print(sogliaUmidita);
    lcd.print("%     ");
   }
   if(primaRiga==3){
    lcd.print("NELLE ORE ");
    if(oraInizio<10) lcd.print("0");
    lcd.print(oraInizio);
    lcd.print("-");
    if(oraFine<10) lcd.print("0");
    lcd.print(oraFine);
   }
}

/**  Creo la funzione che porta l'Arduino in modalità sleep o in delay ogni volta che viene richiamata   **/
void dormi(){
  if(rispEn)  LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
  else  delay(8000);  //Se la mod. risparmio energia è disabilitata crea solo un delay
}
```

5.5) RISULTATO



Il circuito elettronico funziona correttamente con alimentazione a 9V tramite trasformatore e il ruolo del sistema viene svolto correttamente. La scheda Arduino processa correttamente i dati e attiva l'irrigazione al momento giusto. L'elettrovalvola funziona correttamente.

La scheda Real Time Clock rileva e comunica l'orario correttamente. L'LCD visualizza correttamente le informazioni.
Comunicando dal PC alla scheda tramite seriale questa riceve correttamente le istruzioni e le applica correttamente. La scheda funziona correttamente anche in modalità risparmio energetico.
Gli unici problemi rilevati sono marginali, più precisamente:

- In caso la sonda venga estratta dal terreno il circuito di condizionamento interpreta erroneamente il dato in entrata e fornisce alla scheda l'indicazione dell'umidità massima Questa è però una situazione tecnicamente rara da verificarsi, il problema è quindi marginale; questo problema può anzi diventare un punto di forza, in quanto evita che l'irrigazione si accenda erroneamente mentre non stiamo sondando il terreno.
- La fonte di acqua in entrata deve essere in pressione (l'unica situazione in cui crea problemi è se la fonte è una bottiglia o un contenitore, mentre il sistema è progettato per essere fisso)

5.6) LIMITI E FUTURE EVOLUZIONI

Questo progetto presenta essenzialmente solo due limiti:

- Il sistema richiede un collegamento ad una rete idrica (o più in generale ad una fonte di acqua in pressione) in quanto la pressione di una bottiglia non è sufficiente a creare un flusso di acqua regolare.
- L'elettrovalvola durante il funzionamento (ovvero quando fa passare l'acqua) consuma una corrente troppo alta per una batteria classica, per cui la fonte di alimentazione deve essere una batteria sovradimensionata o, preferibilmente, la rette elettrica (chiaramente attraverso un trasformatore da 9V).

Il progetto presenta invece molte possibili evoluzioni:

- La scheda Arduino ha cinque ingressi analogici ancora inutilizzati (pin A0-A5) a cui è quindi possibile collegare altri cinque sensori, permettendo semplicemente modificando poche righe di codice di implementare il controllo di 6 porzioni di terreno.
- Multiplexando sugli ingressi analogici più sensori, sarebbe addirittura possibile ottenere molte più fonti, idealmente infinite.
- Le scheda Arduino ha altre cinque uscite digitali ancora inutilizzate (pin 1,2,4,5,13) a cui è quindi possibile collegare altri circuiti di controllo per altre elettrovalvole, permettendo di controllare separatamente fino a 6 circuiti di irrigazione contemporaneamente.
- Ogni circuito di irrigazione può nativamente supportare due elettrovalvole (grazie ai due

interruttori forniti dal Relè), e idealmente sarebbe possibile pilotare infinite elettrovalvole (semplicemente aggiungendo più BJT e Relè di controllo), fornendo una flessibilità enorme nella realizzazione del sistema di irrigazione.

- Essendo Arduino una scheda programmabile, chiunque possieda le minime conoscenze di programmazione C++ potrà modificare il codice completamente a proprio piacimento, permettendogli di realizzare la migliore soluzione per i propri bisogni senza bisogno di acquistare un altro sistema proprietario.