**NETDUINO E IRRIGAZIONE AUTOMATICA FAI DA TE (TEORIA)**

[[https://secure.gravatar.com/avatar/ca56dc6cc9e8f660385694cbea878f9c?s=28&d=monsterid&r=g](https://www.logicaprogrammabile.it/author/amministratore/) Marco Lai](https://www.logicaprogrammabile.it/author/amministratore/), [7 anni ago](https://www.logicaprogrammabile.it/irrigazione-automatica-netduino/)  5 min read  867

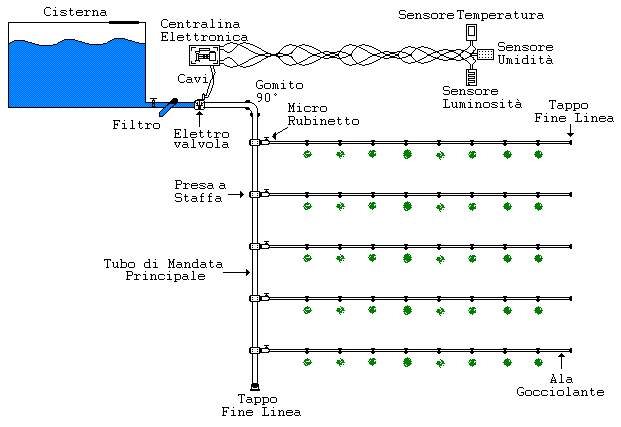
**Come realizzare centralina di micro irrigazione**

Mi è sempre piaciuta l’idea di un orto/giardino automatizzato, magari con **Netduino**, capace di sfruttare l’irrigazione in modo coscienzioso e senza sprechi d’acqua.  
Penso sia uno di quei progetti molto ricercati per la sua evidente utilità pratica, e anche perché la coltivazione e la cura di piante è un istinto innato che ci avvicina alla natura e ci fa sentire bene!

Abito in un piccolo paese dove ogni famiglia possiede un piccolo orto per la coltivazione di verdure e frutti. La tecnica di irrigazione principale è quella a solco (scorrimento), decisamente svantaggiosa e poco efficace per l’eccessivo spreco d’acqua.  
Fortunatamente oggi ci sono molte tecnologie efficienti che permettono l’impiego di sistemi programmabili per la gestione delle attività di irrigazione.

Anche io ho voluto sviluppare il mio sistema a microcontrollore per gestire le fasi di irrigazione di un piccolo orto.  
Per la realizzazione di questo sistema bisogna avere un minimo di esperienza nella coltivazione di piante ed essere consapevoli che ci dovremmo sporcare le mani!!

La tecnica di irrigazione migliore per questo progetto si basa sulla microirrigazione a goccia. E’ necessario quindi procurarci tutto il materiale per l’impianto. Le mie informazioni progettuali sono da considerare di carattere generale, dato che il sistema è influenzato da vari parametri come, la tipologia del terreno (inclinazione, consistenza, composizione, esposizione agli agenti atmosferici) il tipo e la quantità di piantine da coltivare.La figura seguente rappresenta un tipico impianto a goccia automatizzato:



La cisterna contiene l’acqua necessaria ad irrigare le piante, le sue dimensioni variano a seconda della dimensione dell’orto, un valore sufficiente per una autonomia di 2-3 settimane potrebbe essere di 1000 litri.  
Subito dopo troviamo il filtro, questo elemento è essenziale per un impianto di **microirrigazione**in quanto blocca le impurità dell’acqua che col tempo potrebbero otturare i gocciolatori.  
L’elettrovalvola è un rubinetto elettronico che permette, tramite la centralina elettronica, di gestire tutte le fasi d’irrigazione. La centralina riceve input dai sensori di umidità, temperatura e luminosità, in questo modo possiamo sviluppare algoritmi che gestiscono l’irrigazione in funzione dei parametri ambientali.

Il tubo di mandata principale permette di alimentare i vari rami dell’impianto, ogni ramo (ala gocciolante) è sezionabile tramite un minirubinetto, questo permette di effettuare operazioni di manutenzione anche durante l’irrigazione.  
L’ala gocciolante è composta da un tubo, solitamente da 16mm, dove sono collocati i gocciolatori. Ogni gocciolatore irriga una singola pianta.

La realizzazione dell’impianto dipende dalla dimensione del terreno a disposizione e dal numero e tipo di piante da coltivare. Secondo la mia esperienza vi posso dire che per il fabbisogno di una famiglia di 5-6 persone potete coltivare i seguenti prodotti

* 20 piante Pomodoro
* 10 piante Melanzane
* 10 piante Peperoni
* 20 piante Fagiolini
* 10 piante Cetrioli
* 5 piante Zucchine
* 20 piante Lattuga

naturalmente ognuno può aumentare o diminuire queste quantità a seconda delle preferenze e dei gusti personali.  
Per quanto riguarda la distanza tra gocciolatori possiamo dimensionare in questo modo

* 40-50 cm per Pomodori, Melanzane, Peperoni, Fagiolini e Cetrioli
* 60-70 cm per le Zucchine
* 15-20 cm per le Lattughe

Mentre la distanza tra un **ala gocciolante** e la successiva può variare da 40-50 cm a  80-100cm a seconda dello spazio a disposizione, e della comodità che vogliamo avere per passare tra i filari.

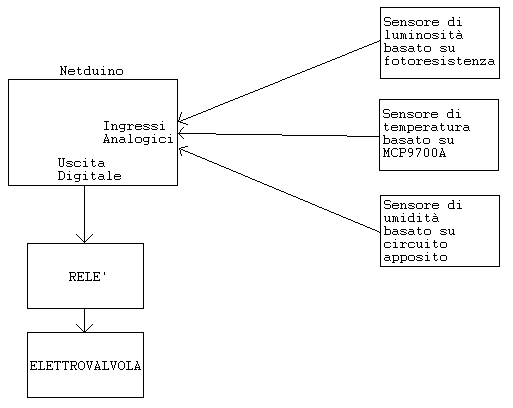
Dopo l’assemblaggio dell’impianto è necessario verificare che tutti i gocciolatori funzionino a dovere e che non ci siano perdite d’acqua dai raccordi di connessione.

La parte che ora bisogna studiare, riguarda la centralina elettronica. Certamente sul mercato esistono vari modelli con prezzi anche di poche decine di euro, che potrebbero farci abbandonare l’idea dell’autocostruzione, ma il nostro scopo è quello di sperimentare e imparare nuovi concetti per poi applicarli in altri ambiti o per accrescere il nostro bagaglio culturale.

Apro una piccola parentesi sull’elettrovalvola: a parte le varie dimensioni e capacità idriche, esistono modelli a 24Vac e a 9Vcc. Volendo creare un impianto autonomo è meglio utilizzare il modello a 9Vcc in modo da eliminare qualsiasi trasformatore. Il costo delle elettrovalvole in continua è purtroppo superiore a quelle in ca (circa 30€).

Il circuito elettronico deve acquisire come input i dati dai sensori e pilotare l’elettrovalvola per un tempo che può essere fisso o variabile a seconda del software che andremmo a sviluppare.  
Come microcontrollore possiamo optare per diverse soluzioni come **Netduino**piuttosto che **Arduino** o **Picmicro**. Certamente un fattore importante è il costo, che si aggira intono ai 30€ per Netduino/Arduino mentre per i Picmicro siamo intorno ai 3-8€.  
In una prima fase possiamo usare Netduino per creare un prototipo, per poi eseguire un porting del progetto su Picmicro in modo da abbattere i costi di realizzazione della centralina.

I sensori del mio prototipo sono tre, un sensore di temperatura analogico tipo **MCP9700a**, un sensore di luminosità basato su **fotoresistenza** e un sensore di umidità realizzato con due puntali da inserire nel terreno.  
Il Netduino acquisisce tramite gli ingressi analogici questi segnali e in base al suo software gestisce un piccolo relè che va a pilotare l’elettrovalvola.  
Nella figura seguente è rappresentato lo schema a blocchi:



Credo che le informazioni di questo articolo siano sufficienti per iniziare a costruire il proprio orto automatizzato. Certamente le nozioni per la realizzazione dell’impianto di microirrigazione sono da approfondire ma con un po di manualità si può apprendere da soli come impostare e realizzare il proprio sistema idrico.  
Per quanto riguarda la parte della realizzazione della centralina elettronica vi rimando al prossimo articolo.

**Leggi Anche Questi Articoli:**

1. [**Realizzazione impianto microirrigazione per orto**](https://www.logicaprogrammabile.it/tutorial-realizzarione-impianto-microirrigazione-domestico/)
2. [**Nuova centralina per impianto di microirrigazione**](https://www.logicaprogrammabile.it/nuova-centralina-impianto-microirrigazione/)
3. [**Capire il duty cycle**](https://www.logicaprogrammabile.it/duty-cycle-guida-definitiva/)
4. [**Pachube, utilizzo con Netduino o Arduino (Introduzione)**](https://www.logicaprogrammabile.it/pachube-utilizzo-netduino-arduino/)
5. [**Aggiornare le schede Netduino alla versione 4.2 del .Net MF**](https://www.logicaprogrammabile.it/aggiornare-netduino-versione-42-net-micro-framework/)

**REALIZZAZIONE IMPIANTO MICROIRRIGAZIONE PER ORTO**

[[https://secure.gravatar.com/avatar/ca56dc6cc9e8f660385694cbea878f9c?s=28&d=monsterid&r=g](https://www.logicaprogrammabile.it/author/amministratore/) Marco Lai](https://www.logicaprogrammabile.it/author/amministratore/), [7 anni ago](https://www.logicaprogrammabile.it/tutorial-realizzarione-impianto-microirrigazione-domestico/)  2 min read  341

**Realizzare impianto di microirrigazione fai da te**

Dopo aver acquistato il materiale necessario alla realizzazione dell’impianto di **microirrigazione** sono passato alla pratica con l’aiuto dei mie familiari. Ho cercato di rispettare lo schema dell’impianto proposto nel [precedente articolo](http://www.logicaprogrammabile.it/irrigazione-automatica-netduino/). Per il momento l’impianto non ha ancora l’**elettrovalvola** perché prima vorrei capire quanta corrente assorbe e fare alcuni calcoli per dimensionare la batteria che dovrà garantire una sufficiente autonomia del sistema.

Per motivi logistici non ho potuto documentare la realizzazione dell’impianto tramite un video. Di seguito alcune foto che illustrano le fasi di realizzazione.

A monte dell’impianto, ho collegato il filtro, fondamentale per un corretto funzionamento dei gocciolatori.

Il filtro è facilmente accessibile, questo mi permette di eseguire delle manutenzioni regolari come la pulizia dell’elemento filtrante.  
La fase successiva è quella di montare le ale gocciolanti al tubo di mandata principale. Nel mio caso il tubo di mandata è stato realizzato precedentemente per ovvi motivi pratici di assemblaggio delle staffe e foratura del tubo.



L’ala gocciolante è inserita nel portagomma del **microrubinetto** collegato alla presa a staffa del tubo di mandata.



L’impianto è composto da 12 tubi di ala gocciolante autocostruita da 16mm



questo un dettaglio del gocciolatore installato nel tubo di 16mm. In questo caso per motivi illustrativi il gocciolatore è rivolto verso l’alto ma in fase di irrigazione il gocciolatore deve essere posto in orizzontale.



In prossimità di ogni gocciolatore ho realizzato una piccola buca per creare la sede dove inserire la pianta



In ogni sede è stato praticato un foro, in direzione del gocciolatore, per inserire la piantina.



Ora inseriamo la piantina nel foro e ricopriamo le radici  con un po di terra



Non ci resta che aprire la valvola e collaudare il sistema. Potete osservare come le gocce bagnano una piccola zona del terreno, quella occupata dalle radici della pianta.



E’ consigliabile controllare ogni singolo gocciolatore per verificarne la corretta erogazione. Se qualche gocciolatore ha un flusso superiore agli altri potete intervenire stringendo la sua ghiera. Assicurarsi una omogenea erogazione, è importante per una irrigazione corretta delle piante.

La parte meno *nobile* del progetto è pronta (a parte l’installazione della elettrovalvola). Nel prossimo articolo presenterò il circuito della centralina formato da uno shield autocostruito e dal nostro amato **Netduino**.

**Leggi Anche Questi Articoli:**

1. [**Netduino, centralina per impianto di microirrigazione**](https://www.logicaprogrammabile.it/come-costruire-fai-da-te-centralina-impianto-microirrigazione/)
2. [**Netduino e irrigazione automatica fai da te (Teoria)**](https://www.logicaprogrammabile.it/irrigazione-automatica-netduino/)
3. [**Nuova centralina per impianto di microirrigazione**](https://www.logicaprogrammabile.it/nuova-centralina-impianto-microirrigazione/)
4. [**Primo ordine dal Fritzing Fab**](https://www.logicaprogrammabile.it/primo-ordine-fritzing-fab/)
5. [**Do It Yourself, la filosofia del fai da te**](https://www.logicaprogrammabile.it/do-it-yourself-diy-fai-da-te/)

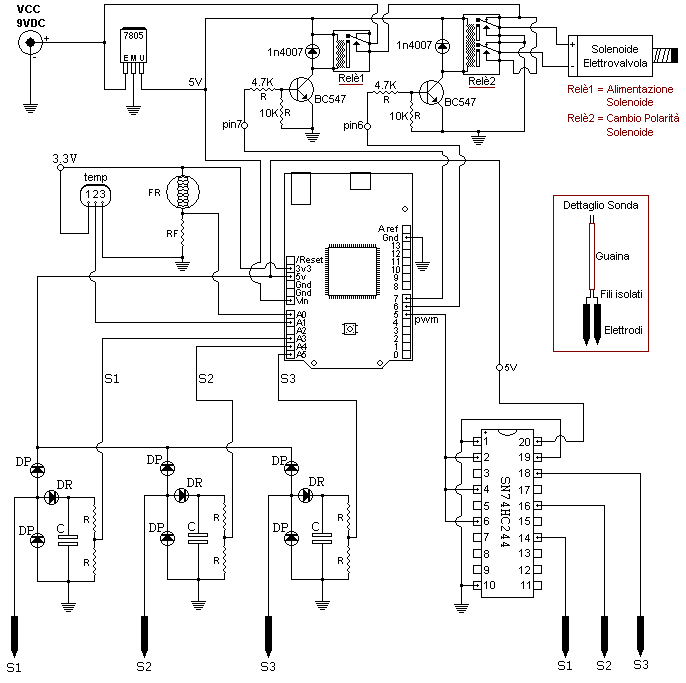
**NETDUINO, CENTRALINA PER IMPIANTO DI MICROIRRIGAZIONE**

[[https://secure.gravatar.com/avatar/ca56dc6cc9e8f660385694cbea878f9c?s=28&d=monsterid&r=g](https://www.logicaprogrammabile.it/author/amministratore/) Marco Lai](https://www.logicaprogrammabile.it/author/amministratore/), [7 anni ago](https://www.logicaprogrammabile.it/come-costruire-fai-da-te-centralina-impianto-microirrigazione/)  7 min read  291

**Come realizzare centralina di microirrigazione con Netduino**

Eccoci giunti alla parte più interessante dell’impianto di microirrigazione proposto negli articoli precedenti. Per il primo prototipo ho deciso di utilizzare il Netduino per eseguire velocemente tutti i test  dell’impianto e per poter adattare il software alle mie esigenze di irrigazione.

Lo schema elettrico del circuito è il seguente



Il circuito sembra complesso ma molti componenti sono stati utilizzati negli articoli precedenti.  
Questo circuito sfrutta maggiormente gli ingressi analogici del Netduino. Sul pin **A0** è collegata la fotoresistenza mentre sul pin **A1**è collegato il sensore di temperatura. Sia la fotoresistenza che il sensore di temperatura sono alimentati con la **3.3V**.  
Gli altri ingressi analogici vengono utilizzati per le tre sonde che verificano l’umidità del terreno. Ho visto alcuni schemi elettrici che, per misurare la conducibilità del terreno, utilizzando una tensione continua. Questo approccio è molto semplice ma può dare problemi di elettrolisi, e conseguente ossidazione delle sonde poste nel terreno, con conseguente imprecisione della misura.  
Per evitare questo problema, possiamo utilizzarne una tensione alternata, in questo caso però dobbiamo successivamente trasformarla in una tensione continua per poterla inviare agli ingressi analogici del Netduino.

Vediamo nel dettaglio il funzionamento della centralina, alcuni argomenti sono stati già esposti e quindi vi rimando alla lettura tramite questi link:

Per il sensore di temperature potete leggere [questo articolo](http://www.logicaprogrammabile.it/mcp9700a-netduino-sensore-temperatura-analogico/).  
Per il sensore di luminosità tramite fotoresistenza potete fare riferimento a [questo articolo](http://www.logicaprogrammabile.it/netduino-usare-ingressi-analogici/).

La parte nuova da descrivere riguarda il **sensore di umidità**. Come ho detto prima, preferisco alimentare le sonde con un segnale alternato; la soluzione migliore è quella di utilizzare la funzione **PWM** del **Netduino**disponibile sul **pin5**.  
Il segnale PWM viene inviato a tre ingressi dell’integrato 74HC244 (Line Driver) per avere un po più di corrente e disaccoppiare il Netduino delle sonde. Il **duty cycle** del pwm è del 50% e la sua frequenza è di 10Khz, naturalmente questi parametri possono essere variati in base alle esigenze e alla tipologia del terreno.

Quindi, il segnale si propaga nel terreno relativamente alla quantità di acqua presente. L’altro polo della sonda riceve questo segnale e lo invia al raddrizzatore che provvederà a fornire una tensione continua proporzionale al livello di umidità del terreno.  
Il raddrizzatore è composto semplicemente da un diodo e da un condensatore. I diodi contrassegnati con il nome **DP** servono a limitare eventuali extra-tensioni captate dalla sonda.  
Il partitore resistivo composto dalle resistenze **R** (da 27Kohm) serve a limitare la tensione sui pin analogici, dato che, il valore massimo applicabile non deve essere superiore a 3.3Vdc.

La parte che pilota il solenoide è composta da due relè, questa scelta può sembrare confusa ma penso che sia quella più economica e per capirla, è necessario conoscere  il funzionamento di una elettrovalvola (vedi riquadro)

***Principalmente esistono due tipi di elettrovalvola, quella monostabile e quella bistabile. L’elettrovalvola monostabile è alimentata con una tensione alternata mentre quella bistabile con una continua.***

***Quelle monostabili possiedono un posizione stabile che è quella di chiusura della valvola, per aprirla dobbiamo alimentarla costantemente, se l’alimentazione viene a mancare la valvola ritorna nella sua posizione stabile. Il vantaggio è che se dovesse mancare l’alimentazione al circuito, mentre l’irrigazione è in atto, la valvola si chiude e quindi non ci sono problemi di allagamento.  
Lo svantaggio è un maggior consumo elettrico poiché le valvole devono essere alimentate per tutto il tempo dell’irrigazione.***

***Quelle bistabili possiedono due stati stabili, uno per la valvola chiusa e uno per la valvola aperta. Alimentando il solenoide la valvola si apre e rimane aperta anche togliendo l’alimentazione. Per chiuderla dobbiamo alimentare il solenoide invertendo le polarità, anche in questo caso togliendo l’alimentazione la valvola rimane chiusa.  
Il vantaggio principale e un consumo minimo in termini elettrici, dato che per aprire e chiudere la valvola basta alimentarla per qualche secondo, questo permette di creare automatismi a batteria .  
Lo svantaggio è che se succede qualche problema al circuito o all’alimentazione, mentre la valvola è aperta, questa rimarrà in questo stato creando problemi di allagamento.***

Nel mio caso, per la realizzazione del circuito, ho utilizzato una piccola basetta mille-fori dove ho alloggiato tutti i componenti del circuito. Per scopi prototipali, potete montare il circuito, anche su una breadboard.

La costruzione delle sonde richiede del comune filo elettrico e due elementi metallici da inserire nel terreno.

Una volta assemblato il circuito e le sonde, possiamo passare allo sviluppo del software. Avendo a disposizione vari input possiamo gestire il nostro **impianto di microirrigazione** adattando diversi algoritmi.

Per una irrigazione efficace dovremmo attivare l’elettrovalvola nel tardo pomeriggio, quando il sole inizia a tramontare. L’irrigazione deve avvenire se le sonde indicano una scarsa umidità del terreno. Il sensore di temperature serve per eseguire, ad esempio, una irrigazione aggiuntiva o prolungata, quando le giornate sono più calde.

Comunque sia, l’algoritmo può variare a seconda delle proprie esigenze, dal tipo di coltura e dalla posizione geografica dell’orto. Quello che mi interessa ora non è creare un algoritmo perfetto ma spiegare come utilizzare la centralina. Ognuno potrà poi, creare e personalizzare il proprio algoritmo.

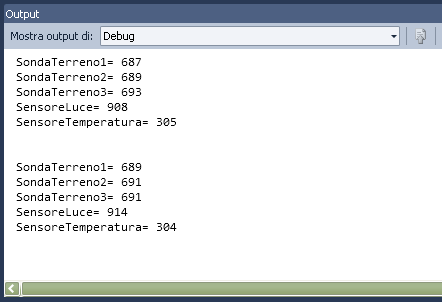
La prima cosa da fare è assicurarsi che tutti i sensori del circuito funzionino a dovere. Il codice seguente acquisisce i segnali dagli ingressi analogici.

***Nota: gli ingressi analogici potrebbero non rispettare la sequenza dello schema elettrico in quanto il prototipo è stato realizzato precedentemente allo schema elettrico***



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | PWM pwmSondeUmidita = new PWM(Pins.GPIO\_PIN\_D5); //uscita pwm su pin5    AnalogInput SondaTerreno1 = new AnalogInput(Pins.GPIO\_PIN\_A0);  AnalogInput SondaTerreno2 = new AnalogInput(Pins.GPIO\_PIN\_A1);  AnalogInput SondaTerreno3 = new AnalogInput(Pins.GPIO\_PIN\_A2);  AnalogInput FotoSensor = new AnalogInput(Pins.GPIO\_PIN\_A3);  AnalogInput TempSensor = new AnalogInput(Pins.GPIO\_PIN\_A4);    //10Khz 50% duty cycle  pwmSondeUmidita.SetPulse(100, 50);    while (true)  {  Debug.Print("SondaTerreno1= " + SondaTerreno1.Read().ToString());  Debug.Print("SondaTerreno2= " + SondaTerreno2.Read().ToString());  Debug.Print("SondaTerreno3= " + SondaTerreno3.Read().ToString());  Debug.Print("SensoreLuce= " + FotoSensor.Read().ToString());  Debug.Print("SensoreTemperatura= " + TempSensor.Read().ToString());    Debug.Print("\n");    //attendo 1 secondo  Thread.Sleep(1000);  } |

Le sonde **S1**, **S2** ed **S3** devono essere distanziate di qualche decina di centimetri cosi che il segnale di una sonda non interferisca con l’altra. Per verificarle ho immerso le sonde in 3 bicchieri di acqua diversi.  
Nella finestra debug otterremo qualcosa di simile



Le sonde per il rilevamento dell’umidità hanno un valore di circa 650-700 quando sono in corto circuito, quando invece non conducono il valore tenderà a 0.  
Il sensore di luminosità ha un valore elevato in presenza di forte luce mentre tenderà a 0 in presenza di poca luce.  
Il sensore di temperatura ha un valore proporzionale alla temperatura misurata, per ottenere il valore in gradi centigradi vi rimando all’[articolo dedicato al sensore](http://www.logicaprogrammabile.it/mcp9700a-netduino-sensore-temperatura-analogico/).

Eccovi un semplice esempio che può darvi un’idea di come utilizzare la centralina



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106 | Int32 LumDato = 0;  Int32 TempDato = 0;  Int32 UmSens1Dato = 0;  Int32 UmSens2Dato = 0;  Int32 UmSens3Dato = 0;    Boolean AttivaIrrigazione = false;  Int32 Ciclo = 0;    //Configurazione PWM  PWM pwmSondeUmidita = new PWM(Pins.GPIO\_PIN\_D5); //uscita pwm su pin5  //Configurazione attuatori (relè)  OutputPort AlimentazioneValvola = new OutputPort(Pins.GPIO\_PIN\_D7, false);  OutputPort PolarizzazioneValvola = new OutputPort(Pins.GPIO\_PIN\_D6, false);  //Configuraziobe Ingressi analogici  AnalogInput SondaTerreno1 = new AnalogInput(Pins.GPIO\_PIN\_A0);  AnalogInput SondaTerreno2 = new AnalogInput(Pins.GPIO\_PIN\_A1);  AnalogInput SondaTerreno3 = new AnalogInput(Pins.GPIO\_PIN\_A2);  AnalogInput FotoSensor = new AnalogInput(Pins.GPIO\_PIN\_A3);  AnalogInput TempSensor = new AnalogInput(Pins.GPIO\_PIN\_A4);    //10Khz 50% duty cycle  pwmSondeUmidita.SetPulse(100, 50);    while (true)  {  //leggo dal sensore fotosensibile  LumDato = FotoSensor.Read();  //leggo dal sensore 9700a  TempDato = TempSensor.Read();  //leggo dalle sonde  UmSens1Dato = SondaTerreno1.Read();  UmSens2Dato = SondaTerreno2.Read();  UmSens3Dato = SondaTerreno3.Read();    //attendi il tramonto  if (LumDato < 200 && AttivaIrrigazione == true)  {  //Questa variabile serve per attivare l'irrigazione  //una sola volta dopo che la luminosità scende sotto 200  AttivaIrrigazione = false;    //controllo l'umidità nelle tre sonde  //Avvia l'irrigazione solo se necessaria (terreno secco)  if (UmSens1Dato < 300 && UmSens2Dato < 300 && UmSens3Dato < 300)  {  //se la temperatura ha un valore elevato irriga  //per un tempo superiore  if (TempDato > 350)  {  //Irrigazione prolungata per alta temperatura    //attivo l'elettrovalvola per un secondo  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);    //l'elettrovalvola è attiva  //attendo 40 minuti prima di disattivarla  for (Ciclo = 0; Ciclo <= 2400; Ciclo++)  {  Thread.Sleep(100);  }    //Inverto le polarità del solenoide  //e poi l'alimento per 1 secondo  PolarizzazioneValvola.Write(true);  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);  PolarizzazioneValvola.Write(false);  }  else  {  //Irrigazione Normale  //attivo l'elettrovalvola per un secondo  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);    //l'elettrovalvola è attiva  //attendo 15 minuti prima di disattivarla  for (Ciclo = 0; Ciclo <= 900; Ciclo++)  {  Thread.Sleep(1000);  }    //Inverto le polarità del solenoide  //e l'alimento per 1 secondo  PolarizzazioneValvola.Write(true);  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);  PolarizzazioneValvola.Write(false);  }  }  }  else if (LumDato > 200)  {  //Rispristina la variabile quando la luminosità  //risale sopra i 200  AttivaIrrigazione = true;  }    Thread.Sleep(500);  } |

Appena alimentiamo il circuito, Netduino esegue il codice creando gli oggetti per l’acquisizione dagli ingressi analogici, per generare il segnale pwm e per comandare i due relè, successivamente esegue un ciclo infinito acquisendo i dati dai sensori ogni 500 millisecondi.

Durante il giorno, in presenza di una buona luminosità, il sistema rimane nella fase di acquisizione; appena la luminosità inizia a diminuire, presumibilmente per il calar del sole, il valore acquisito dal pin analogico **A0**, collegato alla fotoresistenza, diminuisce.  
Quando questo valore scende sotto la soglia di 200 (variabile in funzione del vostro sistema),  Netduino entra nel primo blocco **if**, dato che, **AttivaIrrigazione** è uguale a True.  
Se le tre sonde indicano una bassa umidità del terreno (dato inferiore a 300), Netduino entra nel secondo blocco **if**. A questo punto troviamo un’altra istruzione **if** che permette di eseguire il codice per attivare l’elettrovalvola a seconda della temperatura misurata dal sensore 9700a.  
Quindi se la temperatura è elevata verrà eseguito il primo blocco **if**, altrimenti verrà eseguito il codice del blocco **else**.  
Il codice di questi blocchi è identico, tranne per il ciclo **for**che, per una temperatura elevata, dura maggiormente.

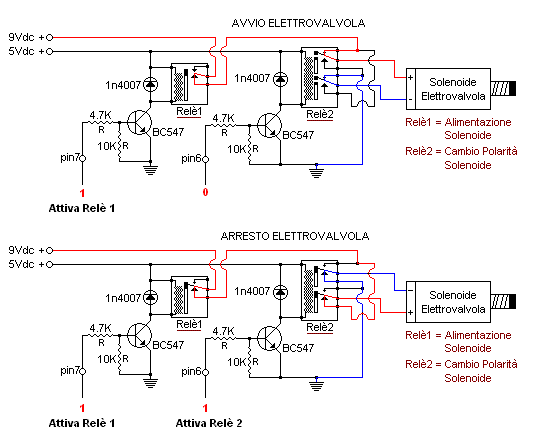
Analizziamo meglio il codice che pilota l’elettrovalvola



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | //Irrigazione Normale  //attivo l'elettrovalvola per un secondo  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);    //l'elettrovalvola è attiva  //attendo 15 minuti prima di disattivarla  for (Ciclo = 0; Ciclo <= 900; Ciclo++)  {  Thread.Sleep(1000);  }    //Inverto le polarità del solenoide  //e l'alimento per 1 secondo  PolarizzazioneValvola.Write(true);  AlimentazioneValvola.Write(true);  Thread.Sleep(1000);  AlimentazioneValvola.Write(false);  PolarizzazioneValvola.Write(false); |

Le istruzioni comandano direttamente i due relè. In condizioni iniziali il **relè2**permette di alimentare il solenoide per aprire l’elettrovalvola. Le prime istruzioni attivano il **relè1,**per alimentare per un secondo il solenoide.  
Il ciclo **for** serve per creare un ritardo software di diversi minuti (tempo di irrigazione).  
Una volta che il ciclo **for** termina, dobbiamo attivare il **relè2** per invertire le polarità di alimentazione del solenoide. Una volta invertite viene attivato il **relè1** per fornire nuovamente tensione al solenoide per circa 1 secondo il quale chiude la valvola, interrompendo l’irrigazione.  
Le ultime istruzioni diseccitano il **relè1** e il **relè2**.

Le illustrazioni seguenti mostrano come avviene la sequenza di azionamento dei relè.



Infine la variabile AttivaIrrigazioneviene utilizzata per irrigare una sola volta, altrimenti il ciclo di irrigazione dura finché le sonde S1, S2 ed S3 rilevano una elevata umidità del terreno.

Il codice è molto esemplificativo, serve per dare una linea guida su come utilizzare la centralina. Nulla vieta di utilizzare un algoritmo differente, o di utilizzare codice più efficiente.

Non ci resta che montare il circuito ed andarlo a sperimentare sul campo.

**Leggi Anche Questi Articoli:**

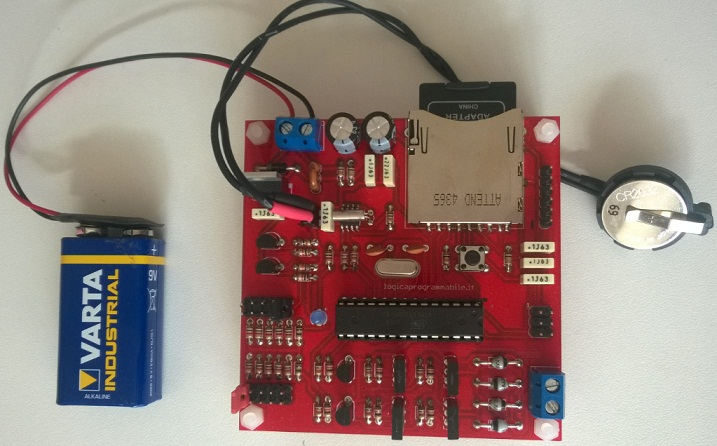
1. [**Nuova centralina per impianto di microirrigazione**](https://www.logicaprogrammabile.it/nuova-centralina-impianto-microirrigazione/)
2. [**Netduino far lampeggiare un led**](https://www.logicaprogrammabile.it/netduino-lampeggiare-led/)
3. [**Realizzazione impianto microirrigazione per orto**](https://www.logicaprogrammabile.it/tutorial-realizzarione-impianto-microirrigazione-domestico/)
4. [**Netduino, linee GPIO configurarle con InputPort e OutputPort**](https://www.logicaprogrammabile.it/netduino-inputport-outputport-tristateport/)
5. [**Come calcolare il crc del tag con Netduino e lettore RFID ID-12**](https://www.logicaprogrammabile.it/calcolare-il-crc-tag-netduino-lettore-rfid-id12/)

<https://www.logicaprogrammabile.it/nuova-centralina-impianto-microirrigazione/>

**NUOVA CENTRALINA PER IMPIANTO DI MICROIRRIGAZIONE**

[[https://secure.gravatar.com/avatar/ca56dc6cc9e8f660385694cbea878f9c?s=28&d=monsterid&r=g](https://www.logicaprogrammabile.it/author/amministratore/) Marco Lai](https://www.logicaprogrammabile.it/author/amministratore/), [7 mesi ago](https://www.logicaprogrammabile.it/nuova-centralina-impianto-microirrigazione/)  6 min read  487

**Come costruire una centralina di irrigazione a goccia**

****

In alcuni miei articoli ho parlato della realizzazione di un impianto di micro irrigazione tramite una [centralina autocostruita](http://www.logicaprogrammabile.it/come-costruire-fai-da-te-centralina-impianto-microirrigazione/). Questa centralina aveva il compito di pilotare una elettrovalvola bistabile utilizzando una [coppia di relè](http://www.logicaprogrammabile.it/come-pilotare-elettrovalvola-bistabile-usando-2-rele/).

Nel tempo ho modificato questo prototipo per renderlo più efficiente e facile da gestire, e soprattutto perché la vecchia centralina aveva dei consumi di corrente elevati e quindi non efficiente con alimentazione a batteria.

Il circuito di questo articolo è attualmente installato nel mio giardino e in quello di alcuni miei amici.

La decisione di realizzare una nuova centralina nasce per via di alcune esigenze:  
la prima è quella dovuta alla scarsità di piogge di questi ultimi anni e di conseguenza la crescente esigenza di irrigare in modo sostenibile utilizzando sistemi ad ala gocciolante o linee autocostruite con micro gocciolatori.  
La seconda è dovuta alla mia passione per il fai da te e per la convinzione che i circuiti autocostruiti, benché inizialmente siano più costosi, possano essere una valida alternativa al modello [usa e getta](https://it.wikipedia.org/wiki/Usa_e_getta) e dell’[obsolescenza programmata,](https://it.wikipedia.org/wiki/Obsolescenza_programmata) infatti conoscendo schema elettrico e codice del progetto possiamo, in caso di rottura di qualche componente, ripararlo o riadattarlo per ulteriori esigenze.

Questa mia nuova centralina permette di utilizzare sia elettrovalvole bistabili sia valvole motorizzate.

Le elettrovalvole bistabili sono consigliate per impianti alimentati a batteria in quanto, tramite un breve impulso elettrico di qualche millisecondo, è possibile aprire o chiudere il passaggio dell’acqua.  
L’uso di una elettrovalvola nel nostro impianto di irrigazione comporta, il più delle volte, l’impiego di un filtro che privi l’acqua delle impurità che potrebbero comprometterne il funzionamento dell’elettrovalvola; le impurità dell’acqua, col tempo, possono ostruire il piccolo canale che porta l’acqua nella camera del solenoide.

Quando l’acqua possiede una torbidità elevata e non possiamo permetterci un sistema filtrante di qualità possiamo optare per una valvola motorizzata (Escludendo però il sistema di irrigazione a gocciolatori ma a irrigazione diretta).  
La valvola motorizzata assorbe più corrente perché il tempo necessario per aprirla o chiuderla è maggiore rispetto all’elettrovalvola (circa3-5 secondi).

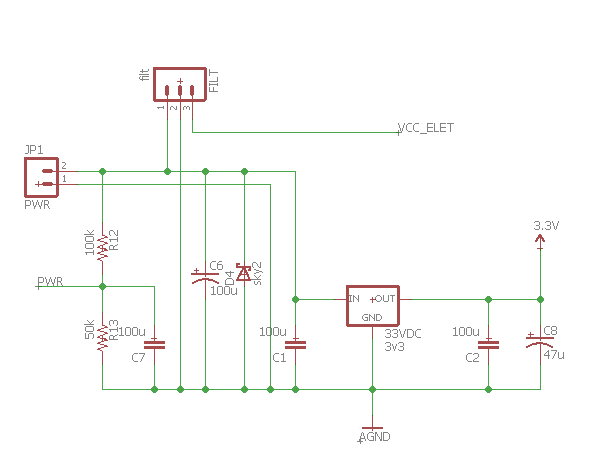
**Caratteristiche nuova centralina di micro irrigazione**

* Basso assorbimento di corrente in standby (circa 0.3uA)
* Open Source e Open Hardware
* Calendario con RTC
* Memorizzazione dati su SD
* Interfaccia seriale
* Programmazione tempi di irrigazione tramite jumper
* Programmazione intervalli di irrigazione tramite jumper
* Led di stato

**Schema elettrico a blocchi**

Lo schema elettrico è stato suddiviso in blocchi. Analizzeremo ognuno di essi con immagini separate per capirne meglio il funzionamento.

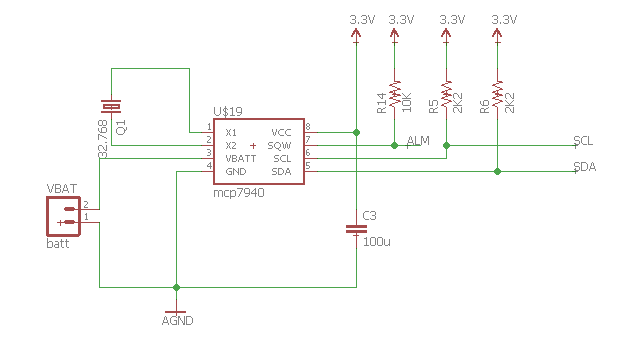
La figura seguente riguarda il sistema di alimentazione del circuito. Il connettore JP1 è collegato alla batteria (da 9 a 12v dc) e permette di alimentare l’elettrovalvola tramite un filtro a larga banda (FILT) e il resto del circuito tramite regolatore a 3.3v (MIC2940A-3.3WT).

Sezione di alimentazione

Il regolatore che ho scelto è un convertitore DC/DC LDO della Microchip. La scelta è ricaduta su questo componente perché assorbe una corrente molto bassa.  
Il componete di nome sky2 è un diodo [transil](https://it.wikipedia.org/wiki/Transil) bidirezionale da 15v che permette di smorzare le extratensioni generate della commutazione del solenoide dell’elettrovalvola.  
Se non adeguatamente filtrate, le extratensioni possono resettare il microcontrollore rendendo il circuito instabile ed inutilizzabile.

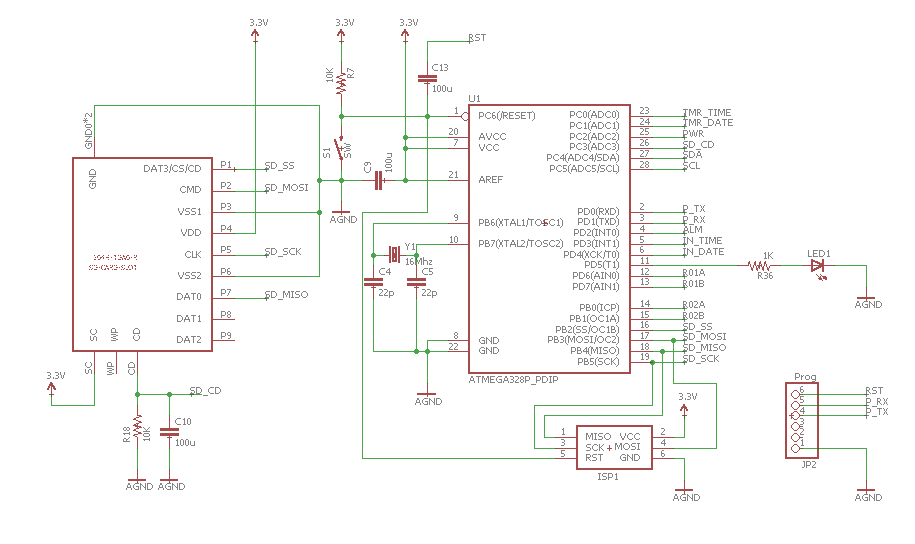
Il partitore di tensione costituito dalle resistenze R12 e R13 serve per monitorare la carica della batteria e di conseguenza arrestare i cicli di irrigazione quando la tensione della batteria non assicura la commutazione dell’elettrovalvola.

Lo schema seguente riguarda invece il circuito RTC:

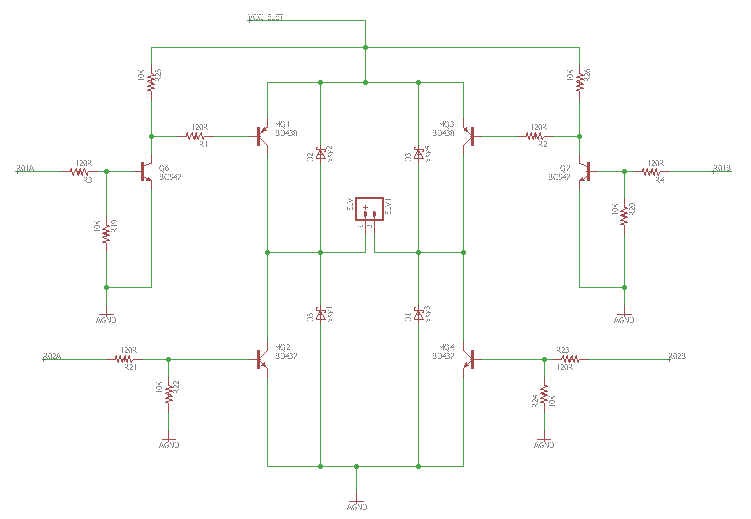
Real Time Clock

questo circuito ha il compito sia di gestire la data e l’ora, ma anche di generare un segnale di allarme (programmato via software), collegato ad un pin di interrupr del microcontrollore, questo sistema mi permette di risvegliare il microcontrollore dallo stato di ibernazione, necessario per garantire bassi assorbimenti del circuito.

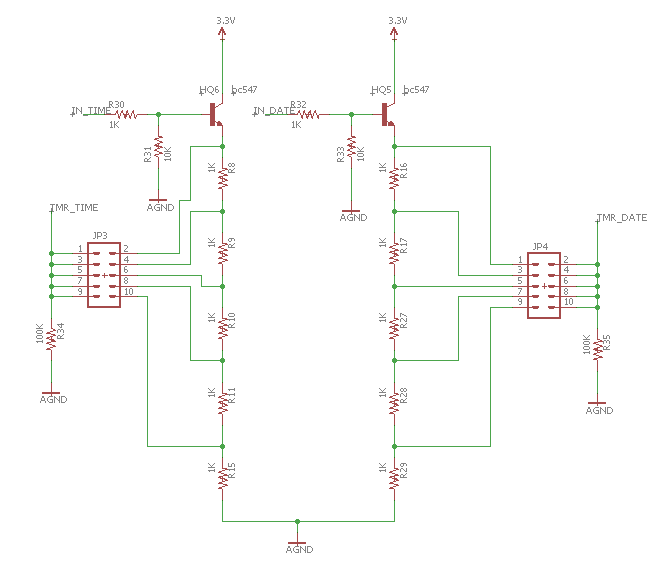
Il cuore della centralina è basato sul microcontrollore ATMEGA328P-PU:



al micro è collegato lo slot SD (la comunicazione avviene usando la SPI). Osserviamo anche il connettore connettore ISP, e il connettore UART per la comunicazione seriale.  
L’elettrovalvola viene pilotata tramite un ponte ad H realizzato con dei transistors; lo schema è il seguente:

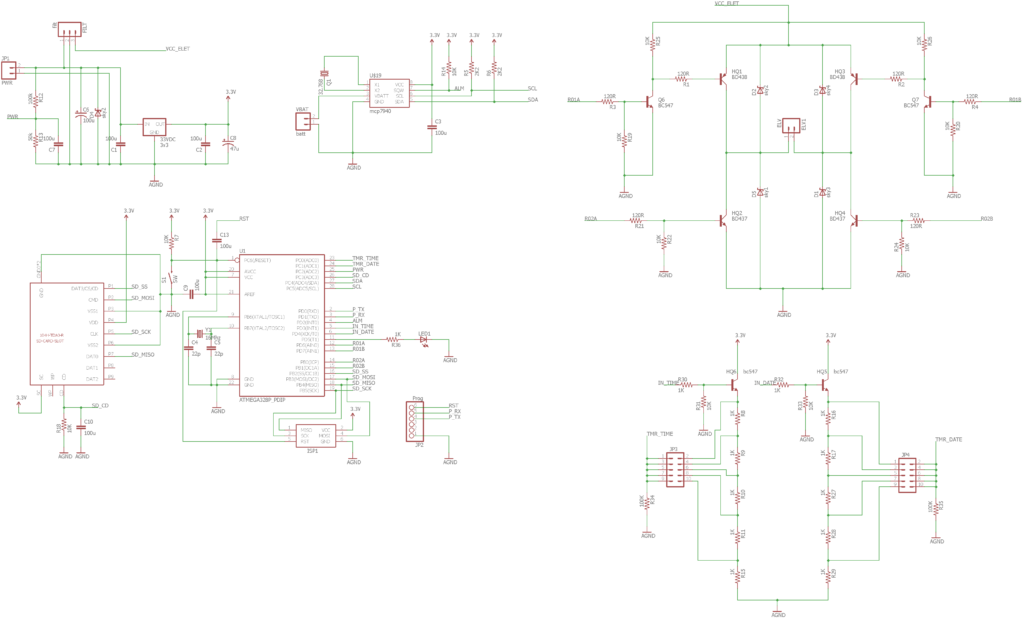
Circuito per pilotare l’elettrovalvola

l’ultimo blocco dello schema elettrico riguarda la selezione del tempo di irrigazione e dell’intervallo di tempo tra una irrigazione e la successiva:

Sezione gestione tempo di irrigazione e gestione intervallo tra una irrigazione e la successiva

Il segnale TMR\_TIME e TMR\_DATE sono collegati a due ingressi analogici del microcontrollore.

Di seguito il circuito elettrico della centralina di micro irrigazione:

[](http://www.logicaprogrammabile.it/wp-content/uploads/2017/09/CentralinaMicroirrigazione.png)Schema Completo centralina micro irrigazione – CLICCARE PER INGRANDIRE

**Descrizione dello schema elettrico**

Il microcontrollore ha il compito di gestire tutte le periferiche collegate alle sue line GPIO. La caratteristica principale di una centralina di questo tipo è il bassissimo assorbimento di corrente; supponendo di alimentare il circuito con una batteria (tipicamente una 9v o nel mio caso 8 batterie stilo alcaline tipo AA) dobbiamo garantire la durata di funzionamento per diversi mesi (naturalmente questo dato dipende dalla capacità della batteria).

Per garantire un basso assorbimento il microcontrollore andrà messo in modalità basso consumo quando non esegue nessuna operazione. Una volta entrato in questa modalità il micro potrà essere risvegliato tramite watchdog oppure tramite interrupt esterno. Nel mio progetto il risveglio è gestito da un interrupt esterno, generato dall’RTC, perché in questo modo posso gestire tempi di standby molto più prolungati rispetto al watchdog.

L’interrupt è generato dalla funzione alarm del modulo rtc MCP7940N. In pratica sul piedino 7 (sqw/MFP) viene generato un cambio di stato logico ogni volta che l’orario dell’ RTC corrisponde all’orario settato sul registro interno ALARM.

Il pilotaggio dell’elettrovalvola avviene utilizzando un ponte ad H costruito con dei transistors discreti. Lo schema permette di essere comandato tramite le uscite digitali del microcontrollore. I diodi schottky posti al centro del ponte H garantiscono l’eliminazione delle sovratensioni generate dalla commutazione del solenoide.

La selezione della durata dell’irrigazione e dell’intervallo tra una irrigazione e la successiva avviene utilizzando dei semplici jumper. I jumper sono posizionati in un partitore di tensione. L’uscita del partitore di tensione è collegata ad un ingresso analogico del microcontrollore. A seconda della posizione del jumper otterremo sull’ingresso analogico tensioni differenti. Dato che il partitore assorbe una cerca corrente, ho utilizzato dei transistor per alimentare il partitore solo quando voglio leggere la posizione dei jumper. I due transistors vengono pilotati da due uscite digitali del microcontrollore.

**Assemblaggio e test della centralina**

Ho dapprima realizzato il progetto utilizzando una basetta millefori, questo perché prima di realizzare il pcb volevo assicurarmi che tutto funzionasse come da progetto. Dopo qualche giorno di test ed aver avuto la certezza che lo schema non presentasse anomalie ho deciso di produrre il pcb.

Dopo aver montato i componenti sul pcb ho sistemato il circuito in una cassetta ermetica per impianti elettrici. Nella cassetta ho inserito anche la batteria. L’unico foro praticato è quello per fare uscire i due fili del solenoide dell’elettrovalvola.

Ho tenuto in osservazione il funzionamento della centralina per circa 2 mesi e mezzo e non ho avuto nessun problema. La posizione del circuito era abbastanza critica perchè rimaneva molto tempo esposta al sole. questa è stata una scelta voluta perché avevo bisogno di sotto porre il circuito a condizioni termiche estreme (giorni con temperature anche superiori ai 40°c). L’unico cosa che ho osservato è una deriva dell’orario dell’RTC, probabilmente perchè non ho eseguito la calibrazione.

In definitiva sono molto contento di questo lavoro perchè ho ottenuto bassi assorbimenti, un costo contenuto del progetto e una grande flessibilità nella programmazione delle irrigazioni.

**Leggi Anche Questi Articoli:**

1. [**Netduino, centralina per impianto di microirrigazione**](https://www.logicaprogrammabile.it/come-costruire-fai-da-te-centralina-impianto-microirrigazione/)
2. [**Realizzazione impianto microirrigazione per orto**](https://www.logicaprogrammabile.it/tutorial-realizzarione-impianto-microirrigazione-domestico/)
3. [**Netduino e irrigazione automatica fai da te (Teoria)**](https://www.logicaprogrammabile.it/irrigazione-automatica-netduino/)
4. [**Come pilotare elettrovalvola bistabile con 2 relè**](https://www.logicaprogrammabile.it/come-pilotare-elettrovalvola-bistabile-usando-2-rele/)
5. [**Costruire la centralina di volo del quadricottero**](https://www.logicaprogrammabile.it/costruire-centralina-volo-quadricottero-arduino-multiwii/)

