

UNIVERSITÀ TELEMATICA INTERNAZIONALE
UNINETTUNO

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in

Ingegneria Informatica

ELABORATO FINALE

in

Elettrotecnica

**REALIZZAZIONE DI UN LABORATORIO
REMOTO DI ELETTRONICA BASATO SU
ARDUINO E RASPBERRY PI**

RELATORE
Prof. Assante Dario

CANDIDATO
Frasca Manuel

ANNO ACCADEMICO 2020/2021

Ringrazio...

*ringrazio il mio relatore, il professore Assante Dario per avermi assistito nella
realizzazione dell'elaborato*

ringrazio i miei genitori per avermi sempre sostenuto e incentivato in ogni mia scelta

ringrazio Claudia la mia compagna che mi aiuta e assiste in ogni mia avversità

Indice

1. Arduino.....	1
1.1 Arduino UNO.....	2
1.1.1 Hardware.....	2
1.1.2 Software.....	3
1.2 Monitor Seriale.....	4
1.3 Plotter Seriale.....	5
2. Raspberry Pi.....	7
2.1 Raspberry Pi 3 B+.....	7
2.2 Pi Camera.....	8
2.2.1 Streaming Web Pi Camera.....	12
3. AD9850.....	15
3.1 Arduino con AD9850.....	15
4. Circuiti Analizzati.....	19
4.1 Raddrizzatore a singola semionda.....	19
4.2 Raddrizzatore a doppia semionda.....	20
4.3 Circuito semplificato con filtro.....	21
4.4 Amplificatore Operazionale.....	22
4.4.1 Amplificatore Operazionale configurazione Invertente.....	23
4.4.2 Amplificatore Operazionale configurazione non Invertente.....	24
5. Oscilloscopio con Arduino.....	26
5.1 Analizzatore di Spettro con Arduino.....	28
6. Node-Red.....	30
6.1 Dashboard di Node-Red.....	31
7. Arduino Ethernet Shield.....	32
7.1 Arduino Web Server.....	33
8. Il Progetto.....	35
Conclusioni e Sviluppi futuri.....	44
Bibliografia.....	45
Sitografia.....	46

Sommario

In questo progetto verrà presentato un laboratorio comandato da remoto, creato con componenti di dimensioni ridotte e a basso costo, utile sia a scopo didattico che in ambito lavorativo.

Nell'elaborato verranno analizzati tutti gli aspetti tecnici delle apparecchiature utilizzate soffermandosi in particolar modo nelle applicazioni basate su Arduino e Raspberry Pi. All'interno dei capitoli sono presenti, oltre alle specifiche tecniche dei singoli componenti, gli sviluppi dei progetti elaborati con Arduino, i circuiti utilizzati e tutte le implementazioni e progetti propedeutici alla realizzazione del progetto finale, spiegato all'interno dell'ultimo capitolo.

I principali strumenti hardware utilizzati per la realizzazione del progetto, sono:

- Due schede a microprocessore (Arduino Uno)
- Ethernet Shield per l'utilizzo di Arduino come Web Server
- Un mini computer (Raspberry Pi 3 B+) per la programmazione e gestione dei dispositivi collegati
- Pi Camera per lo Streaming Web
- Integrato AD9850 per generare segnali

Gli strumenti software, invece, sono i seguenti:

- IDE Arduino (programmazione C/C++)
- Sistema Operativo Raspberry (Raspbian)
- Python (programmazione dello Streaming Web con la Pi Camera)
- Node-Red (scambio di dati tra i vari nodi collegati)
- HTML, JavaScript, PHP e CSS (Interfaccia Web)

Dall'interfaccia Web si seleziona il circuito da analizzare, si imposta la frequenza uscente dal generatore di segnali, si visualizza lo Streaming della Pi Camera ed il grafico ottenuto dall'oscilloscopio. L'interfaccia Web è basata su Arduino che è utilizzato come Web Server e, collegato con l'AD9850, comanda l'uscita del segnale selezionato. Il circuito selezionato verrà analizzato da un secondo Arduino, che svolge la funzione di Oscilloscopio connesso tramite porta seriale con Raspberry Pi. Node-Red mette in ascolto la porta seriale di Raspberry Pi, acquisisce i valori comunicati da Arduino Oscilloscopio e ne crea il grafico che verrà visualizzato nell'interfaccia Web.

Si evidenzia che il laboratorio remoto sviluppato, ha funzionalità limitate dovute alle caratteristiche hardware dei componenti utilizzati per la realizzazione, nonché alla loro economicità. Se questi elementi venissero in futuro implementati, si avrebbe la possibilità di sviluppare un prodotto aziendale con un vasto numero di funzionalità ed una migliore efficienza tecnica.

Il progetto, dunque, può essere una soluzione innovativa per una società proiettata verso un futuro in cui vengono prediletti il lavoro e la didattica a distanza.

Introduzione

I continui mutamenti che stanno investendo la nostra società portano ad un ripensamento continuo del sistema educativo, formativo e lavorativo. Tra i mutamenti del sistema, non si può non annoverare l'innovazione tecnologica, che ha trasformato la nostra società e che ha portato ad una sempre più profonda digitalizzazione del campo formativo e lavorativo. Alla base di questo studio, vi è la volontà di creare un laboratorio comandato da remoto, tramite l'utilizzo di microcontrollori (Arduino) e mini personal computer (Raspberry Pi), comodi da portare, date le ridotte dimensioni e dove è possibile accedervi da qualsiasi luogo ed in qualsiasi momento si voglia, solo avendo una connessione ad Internet.

L'obiettivo di questo progetto è quello di creare un laboratorio efficiente ed economico, che permetta a chiunque di accedervi e lavorarci.

L'elaborato è articolato in 8 capitoli:

Il primo capitolo analizza gli aspetti tecnici del microcontrollore Arduino.

Il secondo capitolo espone l'utilizzo di Raspberry Pi con le sue specifiche tecniche e la visualizzazione tramite Streaming Web della Pi Camera con le sue caratteristiche.

Il terzo capitolo contiene tutte le applicazioni utilizzate con il generatore di segnali AD9850 e l'utilizzo con Arduino

Il quarto capitolo è composto dai vari circuiti elettrici da analizzare, le loro implementazioni, i loro schemi circuitali e le caratteristiche dei componenti adoperati.

Il quinto capitolo studia la possibilità di utilizzare Arduino come Oscilloscopio e come Analizzatore di Spettro per poter visualizzare l'uscita dei circuiti analizzati.

Nel sesto capitolo viene spiegato brevemente il funzionamento dell'applicativo Node-Red e l'utilizzo della Dashboard per il progetto.

Il settimo capitolo affronta l'Ethernet Shield di Arduino e l'implementazione per creare un Web Server basato attraverso l'uso di Arduino.

L'ottavo capitolo infine spiega nel dettaglio tutto il progetto effettuato riprendendo anche le implementazioni e i codici utilizzati nei capitoli precedenti.

1. Arduino e l'IDE

“Arduino comprende una piattaforma hardware per il *physical computing* sviluppata presso l'Interaction Design Institute, un istituto di formazione post-dottorale con sede a Ivrea, fondato da Olivetti e Telecom Italia.

La piattaforma fisica si basa su un circuito stampato che integra un microcontrollore con dei pin connessi alle porte I/O, un regolatore di tensione e, quando necessario, un'interfaccia USB che permette la comunicazione con il computer utilizzato per programmare. A questo hardware viene affiancato un ambiente di sviluppo integrato (IDE) multipiattaforma disponibile per Linux, Apple Macintosh e Windows. Questo software permette anche ai novizi di lavorare con Arduino, in quanto i programmi sono scritti in un linguaggio di programmazione semplice e intuitivo, chiamato Wiring, derivato dal C e dal C++. I programmi in Arduino vengono chiamati *sketch*.

Arduino può essere utilizzato per lo sviluppo di oggetti interattivi *stand-alone* e può anche interagire, tramite un collegamento e un opportuno codice, con software residenti su computer.

La piattaforma hardware Arduino è spesso distribuita agli utenti in versione pre-assemblata, acquistabile su internet o in negozi specializzati. Molte informazioni sull'hardware sono disponibili a chiunque, si tratta di hardware closed source poiché la distinta base e gli schemi circuitali funzionanti non vengono mai rilasciati direttamente dallo staff di Arduino. La community di utenti è riuscita a reperire queste informazioni. Solo grazie a questo sforzo da parte della community di Arduino e alla benevolenza dello staff di Arduino nei confronti di eventuali repliche o plaghi, chi lo desidera può legalmente auto-costruirsi un clone di Arduino o derivarne una versione modificata, scaricando gratuitamente lo schema elettrico e l'elenco dei componenti elettronici necessari. Questa possibilità ha consentito lo sviluppo di prodotti Arduino-compatibili da parte di piccole e medie aziende in tutto il mondo ed è divenuto possibile scegliere tra un'enorme quantità di schede Arduino-compatibili. Tutti questi prodotti sono accomunati dal codice sorgente per l'ambiente di sviluppo integrato e dalla libreria residente che sono resi disponibili e concessi in uso secondo i termini legali di una licenza libera, GPLv2.

Grazie alla base software comune ideata dai creatori del progetto, la comunità Arduino ha potuto sviluppare programmi per connettere a questo hardware con più o meno qualsiasi oggetto elettronico, computer, sensore, display o attuatore. Dopo anni di sperimentazione, è oggi possibile fruire di un database di informazioni vastissimo.”

([https://it.wikipedia.org/wiki/Arduino_\(hardware\)](https://it.wikipedia.org/wiki/Arduino_(hardware)))

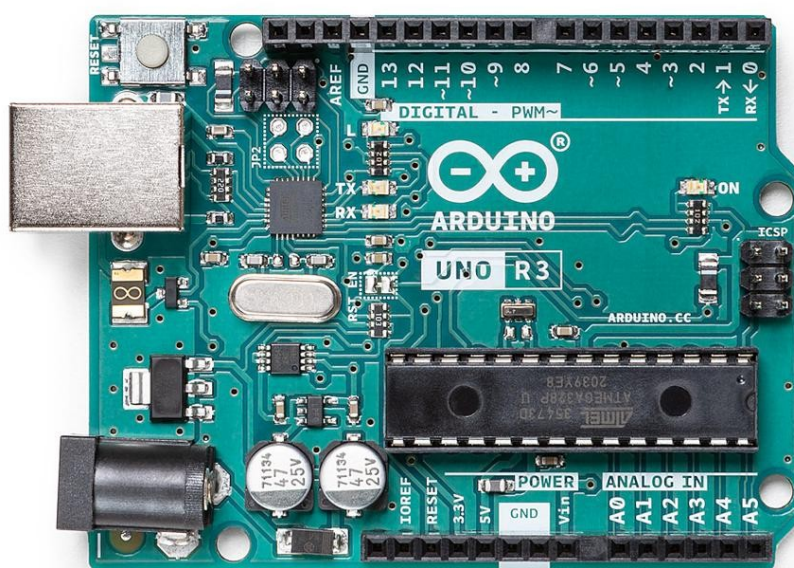
1.1 Arduino UNO

“Arduino Uno è un microcontrollore basato su un processore ATmega328P. Dispone di 14 ingressi/uscite digitali (di cui 6 utilizzabili come uscite PWM), 6 ingressi analogici, un risonatore ceramico da 16 MHz (CSTCE16M0V53-R0), una connessione USB, un jack di alimentazione, un header ICSP e un pulsante di reset.

"Uno" è stato scelto per contrassegnare il rilascio del software Arduino (IDE) 1.0. La scheda Uno e la versione 1.0 del software Arduino (IDE) erano le versioni di riferimento di Arduino, ora evolute nelle ultime versioni.”

(<https://store.arduino.cc/products/arduino-uno-rev3/>)

Figura 1. Arduino



Fonte: <https://store.arduino.cc/products/arduino-uno-rev3/>

1.1.1 Hardware

Di seguito una tabella con le specifiche Hardware di Arduino UNO:

Tabella 1. Specifiche Hardware

MICROCONTROLLORE	ATmega328P
TENSIONE DI ESERCIZIO	5V
TENSIONE DI INGRESSO (CONSIGLIATA)	7-12V
TENSIONE DI INGRESSO (LIMITE)	6-20V
PIN I/O DIGITALI	14 (di cui 6 forniscono uscita PWM)

PIN I/O DIGITALI PWM	6
PIN DI INGRESSO ANALOGICO	6
CORRENTE CC PER PIN I/O	20 mA
CORRENTE CC PER PIN DA 3,3 V	50 mA
MEMORIA FLASH	32 KB (ATmega328P) di cui 0,5 KB utilizzati dal bootloader
SRAM	2KB (ATmega328P)
EEPROM	1KB (ATmega328P)
VELOCITÀ CLOCK	16 MHz
LED_COSTRUITO	13
LUNGHEZZA	68,6 mm
LARGHEZZA	53,4 mm
IL PESO	25 g

Fonte: <https://store.arduino.cc/products/arduino-uno-rev3/>

1.1.2 Software

“Per permettere la stesura del codice sorgente, l'IDE include un editor di testo dotato di alcune particolarità, come il syntax highlighting, il controllo delle parentesi e l'indentazione automatica. L'editor è inoltre in grado di compilare e caricare sulla scheda Arduino il programma eseguibile.

L'ambiente di sviluppo integrato di Arduino è fornito di una libreria software C/C++, chiamata "Wiring" (dall'omonimo progetto Wiring): la disponibilità della libreria rende molto più semplice implementare via software le comuni operazioni di input/output. I programmi di Arduino sono scritti in linguaggio derivato dal C/C++, ma all'utilizzatore, per poter creare un file eseguibile, si richiede solo di definire due funzioni:

void setup ()

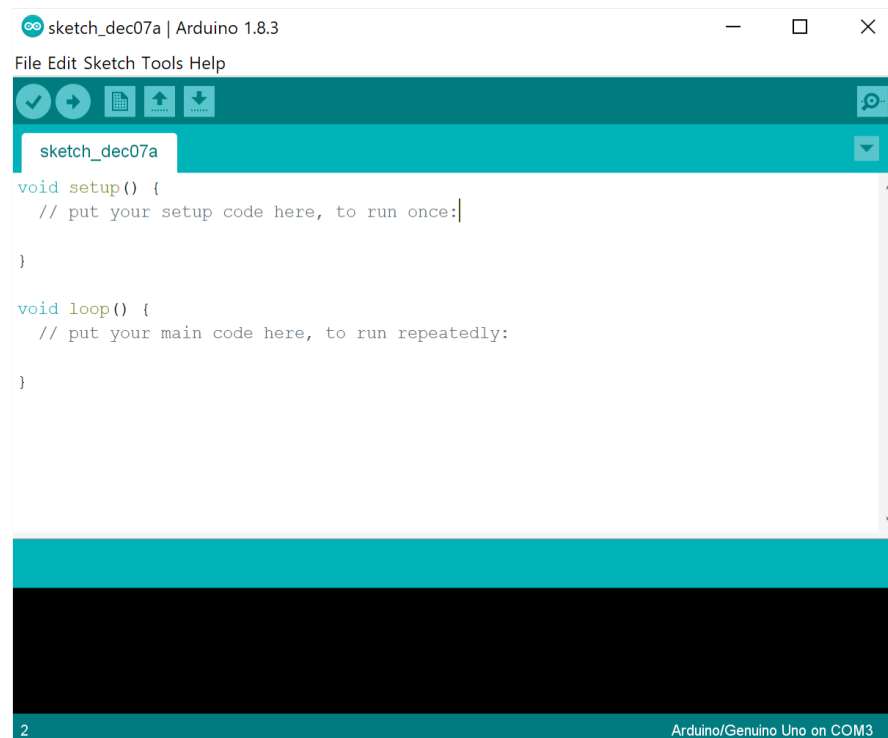
– funzione invocata una sola volta all'inizio di un programma, da utilizzare per i settaggi iniziali che rimarranno invariati durante l'esecuzione;

void loop ()

– funzione invocata ripetutamente, la cui esecuzione si interrompe solo quando si toglie l'alimentazione alla scheda.”

(https://it.wikipedia.org/wiki/Arduino_IDE).

Figura 2. Arduino IDE



Fonte: <https://support.microsoft.com/it-it/topic/caricamento-del-codice-bacheca-e-dell-ide-di-arduino-a9723765-1314-49e0-a69b-bb5c3e1f628d>

1.2 Monitor seriale

Non essendo disponibile un debugger per il microcontrollore Arduino, l'utilizzo del monitor seriale rappresenta l'unica valida alternativa per comprendere i malfunzionamenti del codice scritto.

Il monitor seriale è uno strumento integrato nell'IDE di Arduino per visualizzare i dati mediante comunicazione seriale.

La comunicazione seriale è una modalità di comunicazione tra dispositivi digitali nella quale i bit sono trasferiti lungo un canale di comunicazione uno di seguito all'altro. Nel caso specifico, la comunicazione avviene tra il Computer ed Arduino.

Le istruzioni per inviare messaggi da Arduino al Personal Computer sono due: `Serial.begin` e `Serial.println`.

L'Inizializzazione della comunicazione avviene mediante l'istruzione:

```
Serial.begin(9600);
```

Questa istruzione deve essere inserita all'interno del corpo del `setup` e permette di impostare la comunicazione seriale definendo la velocità della comunicazione in bits per second (baud).

La comunicazione vera e propria avviene invece mediante l'istruzione:

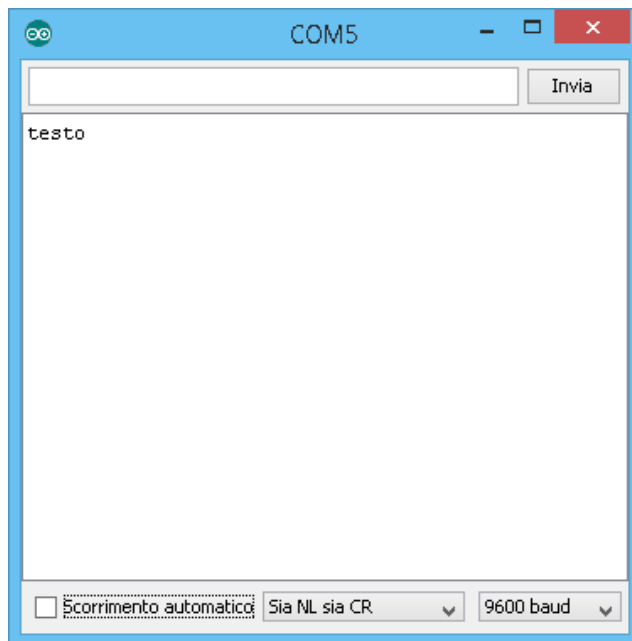
```
Serial.println("Il valore del pulsante risulta:");
```

```
Serial.println(valButton);
```

Nel primo caso viene stampato nel monitor seriale il testo: "Il valore del pulsante risulta:". Mentre nel secondo caso viene stampato il valore della variabile `valButton`.

L'impiego delle `println` permette di capire il valore delle variabili e determinare il corretto funzionamento del circuito.

Figura 3. Monitor Seriale



Fonte: <http://pietrolodi.altervista.org/arduino9/>

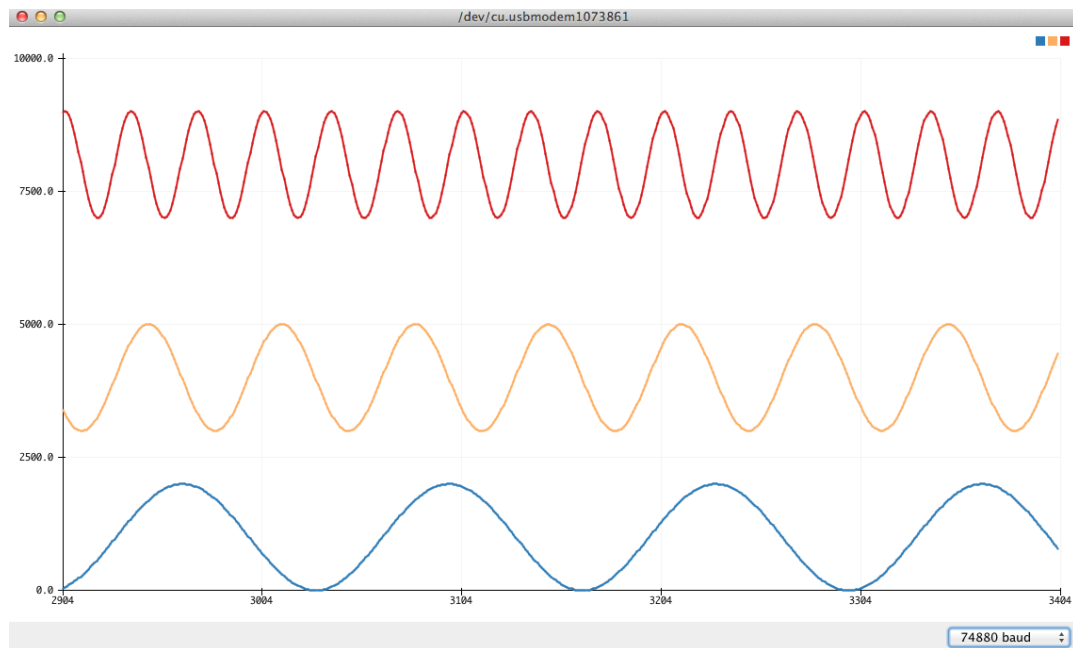
1.3 Plotter seriale

Dalla versione 1.6.7 esiste, all'interno dell'IDE di Arduino, il Plotter Seriale. Analogamente al monitor seriale il Plotter seriale permette la visualizzazione dei dati attraverso un grafico (Figura 4.) Esso è molto utile quando si vuole osservare una particolare grandezza fisica (temperatura, tensione o altro) nel dominio del tempo o se occorre visualizzare, sempre graficamente, una determinata funzione matematica.

Per utilizzare il plotter seriale si devono inviare le informazioni numeriche sulla porta COM, attraverso la funzione `Serial.println()`.

Il plotter seriale risulta particolarmente utile quando si ha a che fare con i dati analogici.

Figura 4. Plotter Seriale



Fonte: <https://www.electronics-lab.com/project/using-arduino-ides-serial-plotter-feature/>

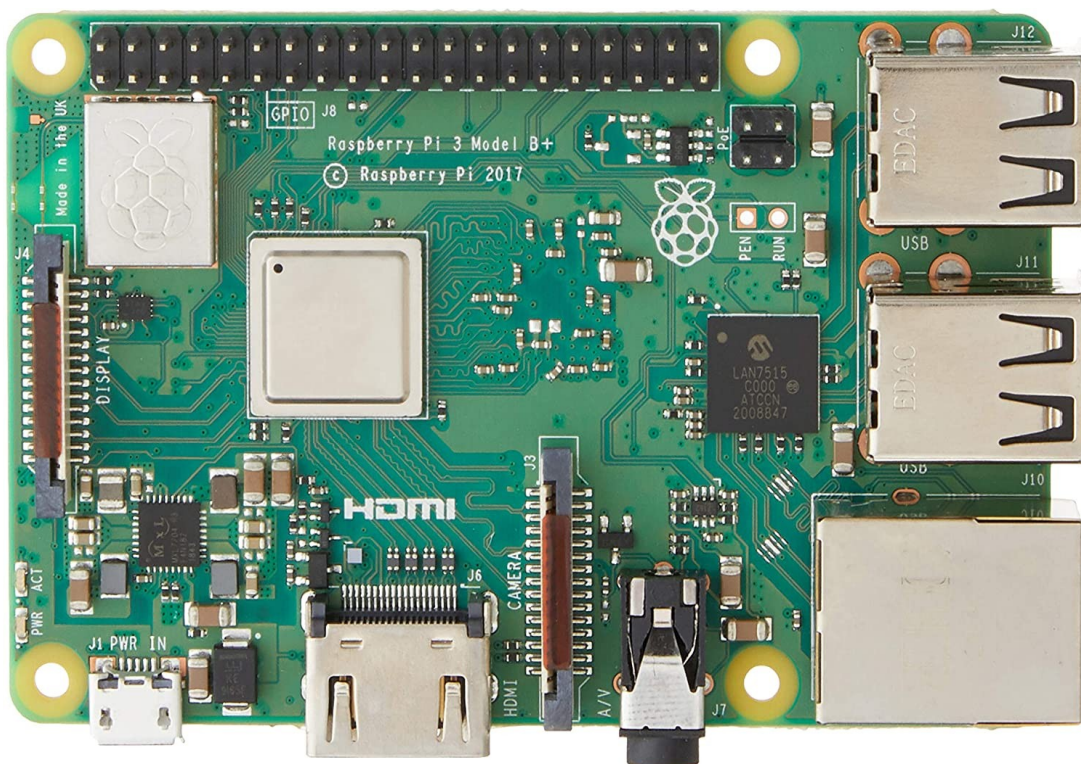
2. Raspberry Pi

In questo capitolo verranno spiegate le caratteristiche di Raspberry, il modello utilizzato e le sue caratteristiche tecniche. Si illustra la Pi Camera attraverso le sue modalità d'uso, le sue specifiche e l'utilizzo con Python per essere trasmessa via web.

2.1 Raspberry Pi 3 b+

Il modello Raspberry Pi 3 B+ (Figura 5.) presenta un processore Broadcom CPU quad-core ARM Cortex-A53 da 1,4 GHz a 64 bit. Questo computer fornisce una connettività wireless dual-band da 2,4 GHz e 5 GHz oltre al Bluetooth 4.2 e BLE. Il modello Raspberry Pi 3 B+ offre una connessione Ethernet veloce (Gigabit Ethernet su USB 2.0) e supporto del Power-over-Ethernet (PoE) con PoE HAT separato. Questo computer fornisce inoltre un ambiente di esecuzione pre-boot (PXE) superiore, un avvio da dispositivi USB Mass Storage e ha un'ottima gestione termica.

Figura 5. Raspberry



Fonte: <https://www.amazon.it/Raspberry-Modello-Piastra-base-verde/dp/B07BFH96M3>

“Le caratteristiche del Raspberry, sono le seguenti:

- Broadcom BCM2837B0, SoC Cortex-A53 da 1,4 GHz e 64 bit

- Connettività wireless dual-band 802.11ac
- Bluetooth 4.2
- Ethernet veloce (Gigabit Ethernet su USB 2.0)
- Supporto PoE (con PoE HAT separato)
- Rete PXE potenziata
- Avvio da dispositivi USB Mass Storage
- Migliore gestione termica

Le specifiche tecniche più salienti possono essere così riassunte:

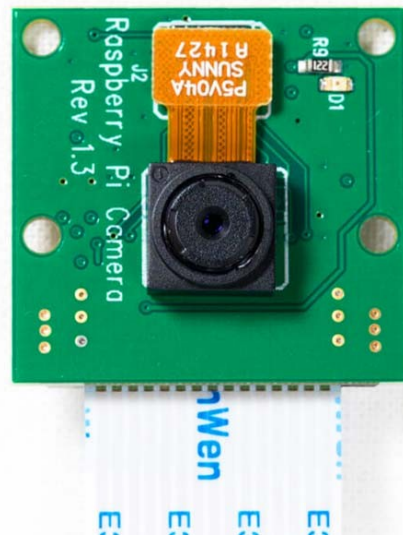
- 1 GB di memoria SDRAM LPDDR2
- Alimentazione CC di ingresso da 5 V/2,5 A (micro USB)
- Connettività wireless:
 - o LAN wireless da 2,4 GHz e 5 GHz IEEE 802.11.b/g/n/ac e Bluetooth 4.2/BLE
 - o Gigabit Ethernet su USB 2.0 (resa massima a 300 Mbps)
 - o 4 porte USB 2.0
- Header con GPIO a 40 pin
- Uscita video HDMI full-size
- Uscita audio stereo a 4 poli e porta video composito
- Porta CSI per il collegamento di una fotocamera Raspberry Pi
- Porta DSI per il collegamento di un display touchscreen Raspberry Pi
- Porta micro SD per caricare il sistema operativo e conservare i dati
- Temperatura di funzionamento: da 0 °C a +50 °C
- Dimensioni: 120 mm x 75 mm x 34 mm
- Peso: 75 g

(<https://www.mouser.it/new/raspberry-pi/raspberry-pi-3-bplus/>)

2.2 Pi Camera

La Pi Camera (Figura 6.) è una smart camera, date le sue piccole dimensioni può essere comodamente fissata ovunque. È dotata inoltre di un'ottima risoluzione, utilizzata con Raspberry e programmabile in Python, è in grado di rilevare movimenti, persone, oggetti, etc., e può anche essere utilizzata come strumento AI (artificial intelligence).

Figura 6. Pi Camera



Fonte: <https://lorenzocasaburo.it/wp-content/uploads/2018/05/Copertina-Pi-Camera.jpg>

Di seguito, si riportano le tabelle delle specifiche tecniche della Pi Camera.

Tabella 2. Specifiche Hardware

Dimensioni	Circa $25 \times 24 \times 9$ mm
Peso	3g
Risoluzione fissa	5 Megapixel
Modalità video	1080p30, 720p60 e 640×480 p60/90
Integrazione Linux	Driver V4L2 disponibile
API di programmazione C	OpenMAX IL e altri disponibili
Sensore	OmniVision OV5647
Risoluzione del sensore	2592×1944 pixel
Area dell'immagine del sensore	$3,76 \times 2,74$ mm
Dimensione pixel	$1,4 \mu\text{m} \times 1,4 \mu\text{m}$
Dimensione ottica	1/4"

Equivalente a un obiettivo SLR full-frame	35 mm
Rapporto segnale/rumore	36dB
Gamma dinamica	67 dB @ 8x guadagno
Sensibilità	680 mV/lux-sec
Corrente oscura	16 mV/sec a 60 C
Bene capacità	4.3 Ke-
Messa a fuoco fissa	1 m all'infinito
Lunghezza focale	3,60 mm +/- 0,01
Campo visivo orizzontale	53,50 +/- 0,13 gradi
Campo visivo verticale	41,41 +/- 0,11 gradi
Rapporto focale (F-Stop)	2.9

Fonte: <https://www.raspberrypi.org/documentation/accessories/camera.html>

Tabella 3. Caratteristiche hardware

A disposizione	Implementato
Correzione dell'angolo del raggio principale	sì
Controllo automatico dell'esposizione (AEC)	No - fatto dall'ISP invece
Bilanciamento del bianco automatico (AWB)	No - fatto dall'ISP invece
Calibrazione automatica del livello del nero (ABLC)	No - fatto dall'ISP invece
Rilevamento automatico della luminanza a 50/60 Hz	No - fatto dall'ISP invece
Frame rate fino a 120 fps	Massimo 90 fps. Limitazioni sulla dimensione dei fotogrammi per i frame rate più elevati (solo VGA per oltre 47 fps)

A disposizione	Implementato
Controllo dimensione/posizione/peso a 16 zone AEC/AGC	No - fatto dall'ISP invece
Specchia e capovolgi	sì
Ritaglio	No - fatto invece dall'ISP (tranne la modalità 1080p)
Correzione delle lenti	No - fatto dall'ISP invece
Cancellazione pixel difettosa	No - fatto dall'ISP invece
Dati RGB RAW a 10 bit	Sì - conversioni di formato disponibili tramite GPU
Supporto per LED e modalità flash strobo	Flash LED
Supporto per la sincronizzazione dei fotogrammi interni ed esterni per la modalità di esposizione dei fotogrammi	No
Supporto per binning 2×2 per un migliore SNR in condizioni di scarsa illuminazione	Qualsiasi risoluzione di output inferiore a 1296×976 utilizzerà la modalità 2×2 binned
Supporto per sottocampionamento orizzontale e verticale	Sì, tramite binning e skip
Anello di blocco di fase su chip (PLL)	sì
Interfaccia standard SCCB seriale	sì
Interfaccia di uscita parallela della porta video digitale (DVP)	No
Interfaccia MIPI (due corsie)	sì
32 byte di memoria integrata programmabile una volta (OTP)	No
Regolatore integrato da 1,5 V per l'alimentazione del core	sì

Fonte: <https://www.raspberrypi.org/documentation/accessories/camera.html>

Tabella 4. Caratteristiche del software

Formati immagine	JPEG (accelerato), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888
Formati video	raw h.264 (accelerato)
Effetti	negativo, solarise, posterize, lavagna bianca, lavagna, schizzo, denoise, rilievo, pittura a olio, tratteggio, gpen, pastello, acquerello, pellicola, sfocatura, saturazione
Modalità di esposizione	auto, notte, anteprima notturna, retroilluminazione, riflettore, sport, neve, spiaggia, molto lungo, fps fissi, antishake, fuochi d'artificio
Modalità di misurazione	media, spot, retroilluminato, matrice
Modalità di bilanciamento bianco automatico	spento, automatico, sole, nuvola, ombra, tungsteno, fluorescente, incandescente, flash, orizzonte
Trigger	Pressione tasti, segnale UNIX, timeout
Modalità extra	demo, burst/timelapse, buffer circolare, video con vettori di movimento, video segmentato, anteprima live su modelli 3D

Fonte: <https://www.raspberrypi.org/documentation/accessories/camera.html>

2.2.1 Streaming Web Pi Camera

Per questo progetto è necessario utilizzare la Pi Camera di Raspberry per trasmettere via Web lo streaming del laboratorio remoto.

“Lo streaming di video sul Web è un mondo abbastanza articolato. Al momento, non esistono ancora standard video universalmente supportati da tutti i browser Web su tutte le piattaforme. Inoltre, HTTP è stato originariamente progettato come protocollo one-shot per servire pagine web. Dalla sua invenzione, sono state aggiunte varie funzioni per soddisfare i suoi casi d'uso sempre crescenti (download di file, ripresa, streaming, ecc.) Ma resta il fatto che al momento non esiste una soluzione "semplice" per lo streaming video.

Per questo progetto verrà utilizzato il formato video MJPEG. Il seguente script utilizza il modulo integrato di Python http.server per creare un semplice server di streaming video:

```
import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server
```

```

PAGE="""\
<html>
<head>
<title>picamera MJPEG streaming</title>
</head>
<body>
<h1>PiCamera MJPEG Streaming</h1>

</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))

```

```

        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()

```

Una volta che lo script è in esecuzione, inserendo l'indirizzo http://IP_Address:8000/ sul browser web si visualizza lo streaming della Pi Camera.”

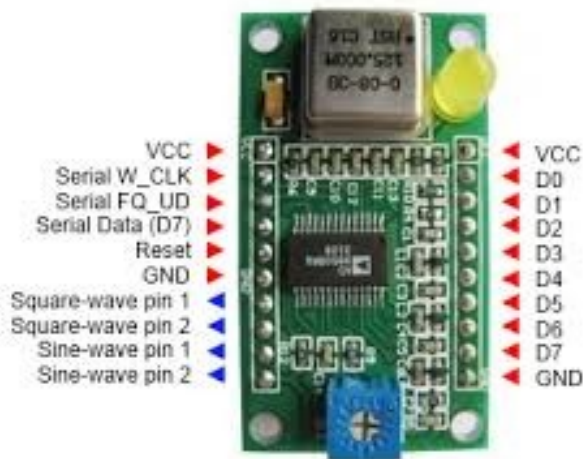
(<https://picamera.readthedocs.io/en/release-1.13/recipes2.html#web-streaming>)

3. AD9850

Nel seguente capitolo verranno spiegate le caratteristiche del componente elettronico AD9850 (Figura 7.) e la sua applicazione insieme ad Arduino per generare segnali.

L'AD9850 è un generatore di segnali dalla frequenza variabile dai 0 ai 40MHz. L'integrato dispone di 4 uscite: 2 ad onda sinusoidale e 2 ad onda quadra. Tali uscite sono programmabili via software attraverso i pin di ingresso Serial W_CLK, Serial FQ_UD, Serial Data (D7) e Reset.

Figura 7.AD9850



Fonte: <https://opencorelibs.com/lib/ad9850spi>

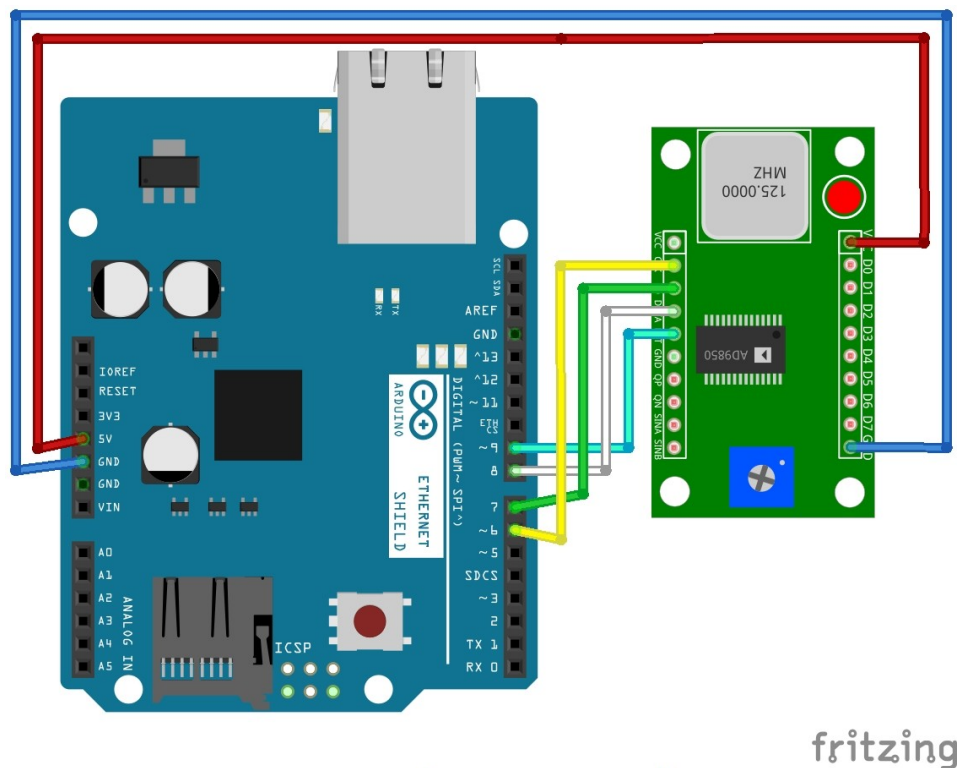
L'uscita che interesserà questo progetto sarà quella sinusoidale. La tensione generata dall'uscita ha un'ampiezza di circa 1,2 Volt ed un valor medio di 600mV. Mentre la frequenza di interesse sarà compresa tra 0 e 2000 Hz.

3.1 Arduino con AD9850

Attraverso lo Sketch di Arduino si può comandare l'uscita dell'AD9850.

Di seguito verrà studiata la configurazione circuitale di Arduino con l'AD9850 (Figura 8.), la configurazione di Arduino con potenziometro e pulsanti (Figura 9.) ed il programma per generare un'onda sinusoidale con frequenza e fase variabili tramite l'uso di un potenziometro (frequenza) e due pulsanti (fase).

Figura 8. Arduino con AD9850

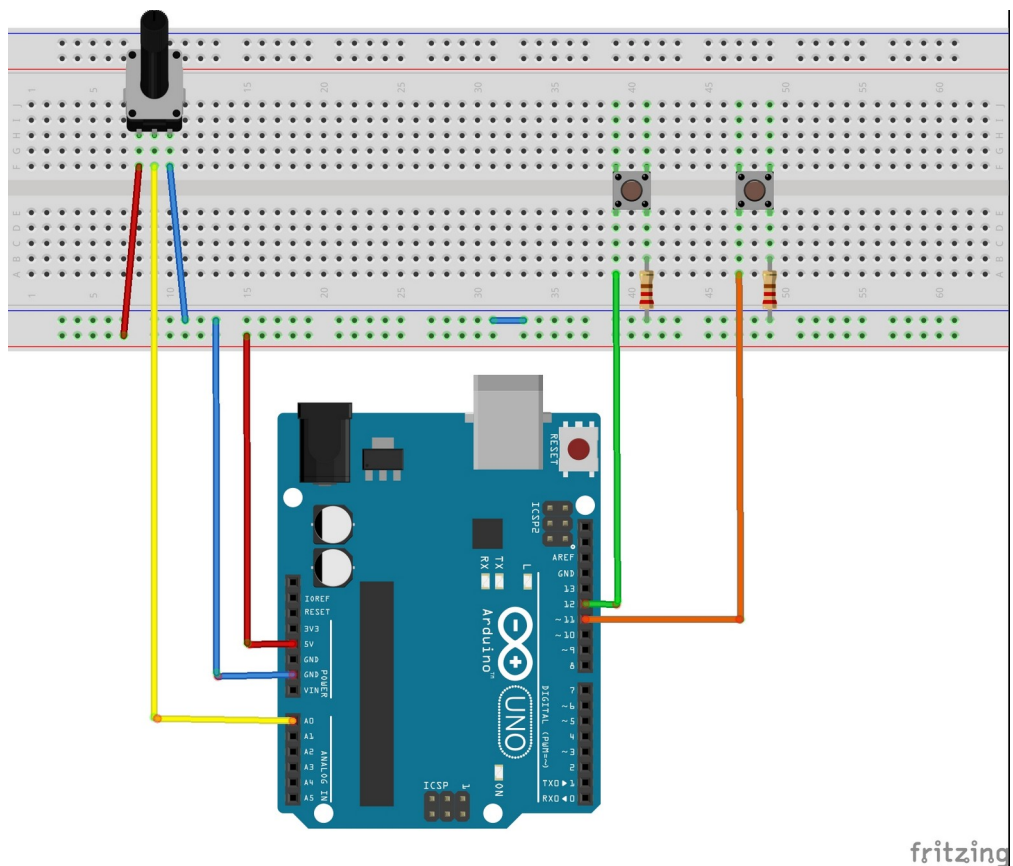


Collegato Arduino con l'AD9850 come mostrato in Figura 8. si prosegue collegando il potenziometro e i due pulsanti come illustrato in Figura 9.

Il potenziometro, attraverso il collegamento dal piedino centrale all'ingresso Analogico (A0) di Arduino, invia un valore di tensione compreso tra 0 e 5V che da Arduino viene interpretato, tramite il convertitore analogico digitale, in un valore compreso tra 0 e 1023.

Si trasforma poi questo dato in una frequenza variabile da 0 Hz a 4000 Hz da comandare all'uscita dell'AD9850. Mentre attraverso la pressione di uno dei due pulsanti si può comandare una variazione di fase impostata a priori dal programma in esecuzione (nel programma sotto riportato con la pressione di un pulsante si avrà una variazione di $+45^\circ$ mentre con la pressione dell'altro di -45°). Se dovesse interessare una variazione di fase più specifica si possono sostituire i due pulsanti con un potenziometro.

Figura 9. Arduino con potenziometro e pulsanti



Il programma in esecuzione ad Arduino è il seguente:

```
#include <AD9850.h> //include la libreria dell'AD9850
```

```
int p1 =12; //primo pulsante collegato al pin digitale 12
```

```
int p2 =11; //secondo pulsante collegato al pin digitale 11
```

```
double freq = 10.0; //frequenza iniziale
```

```
int phase = 0; // fase iniziale
```

```
void setup() {
```

```
Serial.begin(115200); //setta la seriale a 115200 Bound
```

```
DDS.begin(6, 7, 8, 9); //funzione della libreria AD9850 inizializza i pin 6,7,8,9 di Arduino  
che sono rispettivamente collegati con i pin W_CLK, FQ_UD, Serial Data, Reset  
dell'AD9850
```

```
DDS.calibrate ( 124999500 ); //funzione della libreria AD9850 inizializza un valore  
standard per la calibrazione
```

```
DDS.up(); //funzione che inizializza l'AD9850
```

```
DDS.setfreq(freq, phase); // funzione setta l'uscita con frequenza e fase impostate inizialmente
```

```
pinMode(p1,INPUT); // setta il pin 12 dove è collegato il primo pulsante come pin di INPUT
```

```
pinMode(p2,INPUT); // setta il pin 11 dove è collegato il secondo pulsante come pin di INPUT
```

```
}
```

```
void loop() {
```

```
if(digitalRead(p1)==1){ // se il primo pulsante viene premuto la fase aumenta di 45°  
    phase=phase+45;  
}
```

```
if(digitalRead(p2)==1){ // se il secondo pulsante viene premuto la fase diminuisce di 45°  
    phase=phase-45;  
}
```

```
if(phase>=360 or phase<=-360){ // se la fase raggiunge o supera i |360°| ritorna a zero  
    phase=0;  
}
```

```
freq=map(analogRead(A0),0,1024,0,4000); //nel pin A0 è collegato il potenziometro.  
Essendo a 10 bit il convertitore analogico digitale di Arduino si ottengono in ingresso da 0  
ad un massimo 1024 valori. La funzione map converte questo range in uno a scelta che in  
questo caso è il minimo e il massimo della frequenza che si desidera ottenere. La frequenza  
varierà quindi da 0 a 4000 Hz
```

```
DDS.setfreq(freq, phase); //funzione che setta frequenza e fase al valore attuale
```

```
Serial.println(freq); //stampa sulla porta seriale il valore della frequenza
```

```
Serial.println(phase); //stampa sulla porta seriale il valore della fase
```

```
delay(100); // attende 100 millisecondi
```

```
}
```


4 Circuiti analizzati

Nel progetto, come circuiti da analizzare, si studieranno i circuiti di diodi con configurazione di raddrizzatore a singola semionda e raddrizzatore a doppia semionda o ponte di Graetz. Si è utilizzato anche un amplificatore operazionale per aumentare la tensione uscente dal generatore di frequenze. Infine in uscita si è adoperato un filtro RC con un diodo Zener per proteggere l'ingresso analogico di Arduino da tensioni indesiderate.

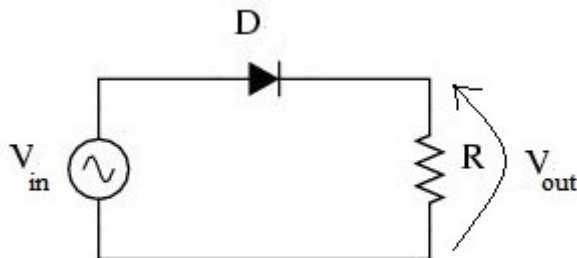
4.1 Raddrizzatore a singola semionda

Il circuito raddrizzatore a singola semionda (Figura 10.) consiste in una tecnica dove si preleva una tensione alternata AC e si riportano al carico solo le semionde positive (Figura 11.) o solo le semionde negative. La realizzazione di tale circuito richiede i seguenti componenti:

- 1 trasformatore 220V – 5V
- 1 resistenza da 10kOhm
- 1 diodo raddrizzatore

Di seguito lo schema circuitale:

Figura 10. Raddrizzatore a singola semionda

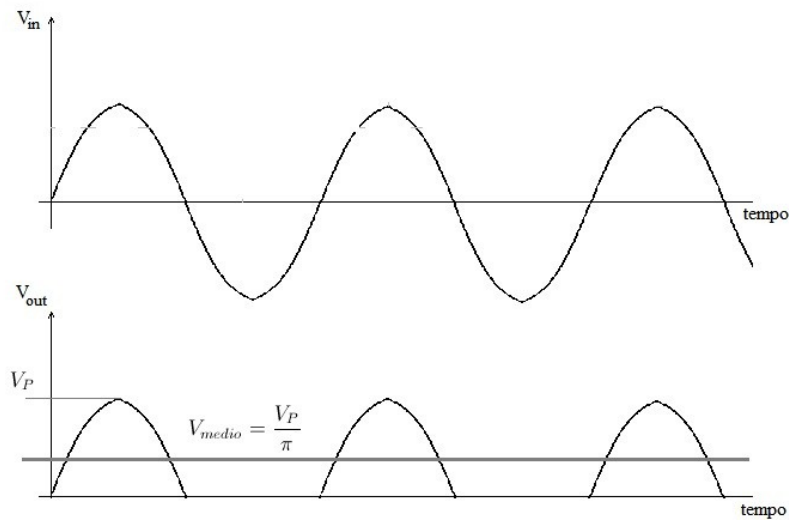


Fonte: <http://www.elemania.altervista.org/diodi/pn/pn5.html>

La tensione 220V alternata è prelevabile dalle prese domestiche, il trasformatore abbassando la tensione a 5V alternata darà la possibilità di lavorare con il circuito (per non guastare i componenti e lavorare in sicurezza).

All'uscita del trasformatore collegando un diodo ed una resistenza da 10kOhm in serie, ai capi del resistore misurando la tensione si avrà l'uscita in Figura 11:

Figura 11. Uscita del segnale



Fonte: <http://www.elemania.altervista.org/diodi/pn/pn5.html>

4.2 Raddrizzatore a doppia semionda

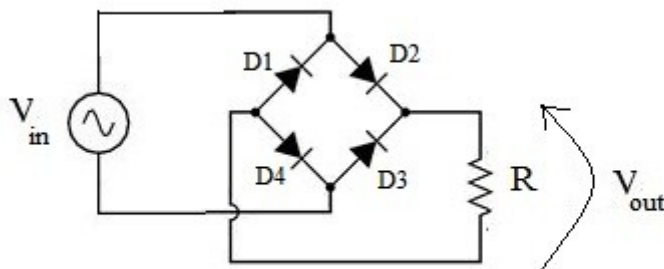
Mentre nel circuito a singola semionda si riportano al carico solo le semionde positive il circuito a doppia semionda (Figura 12.) utilizza una tecnica per il quale prelevando una tensione alternata AC si riporta al carico oltre alla semionda positiva anche la semionda negativa invertita di segno.

La realizzazione di tale circuito richiede i seguenti componenti:

- 1 trasformatore 220V – 5V
- 1 resistenza da 10kOhm
- 4 diodi raddrizzatori

Di seguito lo schema circuitale in Figura 12:

Figura 12. Raddrizzatore a doppia semionda



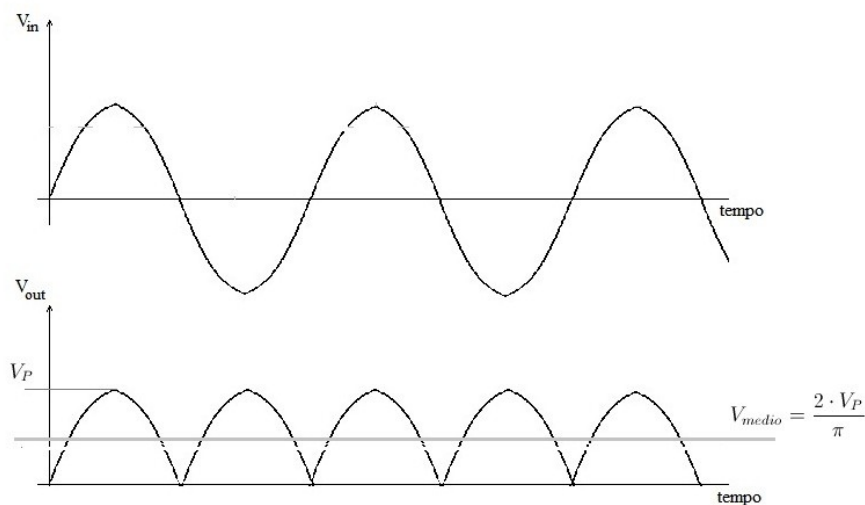
Fonte: <http://www.elemania.altervista.org/diodi/pn/pn6.html>

La tensione 220V alternata è prelevabile dalle prese domestiche, il trasformatore abbassando la tensione a 5V alternata darà la possibilità di lavorare con il circuito (per non guastare i componenti e lavorare in sicurezza).

All'uscita del trasformatore si collegano quattro diodi posti come in Figura 12.

Questa particolare struttura di collegamento di quattro diodi si dice a ponte o, più precisamente, a ponte di Graetz. Ai capi della resistenza misurando la tensione si avrà l'uscita in Figura 13:

Figura 13. Uscita del segnale



Fonte: <http://www.elemania.altervista.org/diodi/pn/pn6.html>

4.3 Circuito semplificato con filtro

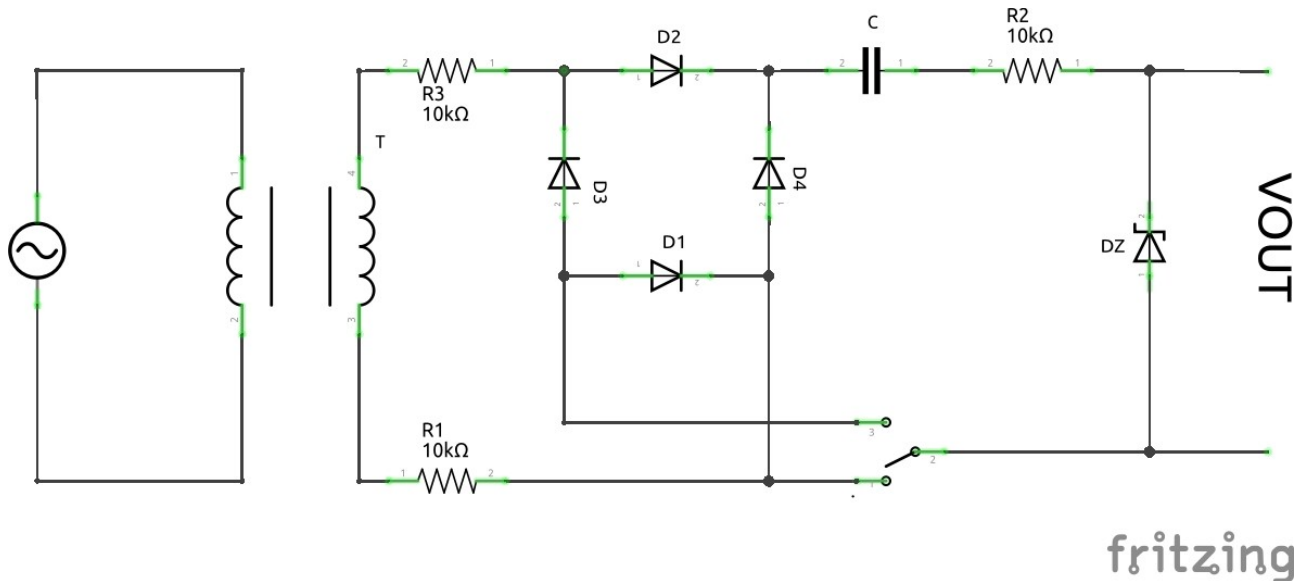
I due circuiti, a singola e doppia semionda, possono essere ridotti ad un solo circuito. Così facendo si utilizzano un numero minore di componenti, avendo comunque lo stesso risultato.

La realizzazione di tale circuito richiede i seguenti componenti:

- 1 trasformatore 220V – 5V
- 3 resistenze da 10kOhm
- 4 diodi raddrizzatori
- Condensatore preferibilmente da 1 micro Farad
- Diodo Zener da 4,7V

Si svilupperà un circuito come in Figura 14:

Figura 14. Circuito con filtro



In questo circuito utilizzando il ponte di Graetz è possibile analizzare sia il raddrizzatore a doppia semionda che il raddrizzatore a singola semionda uscente da VOUT grazie allo switch presente a massa.

La VOUT sarà l'uscita da analizzare che andrà collegata ad un oscilloscopio.

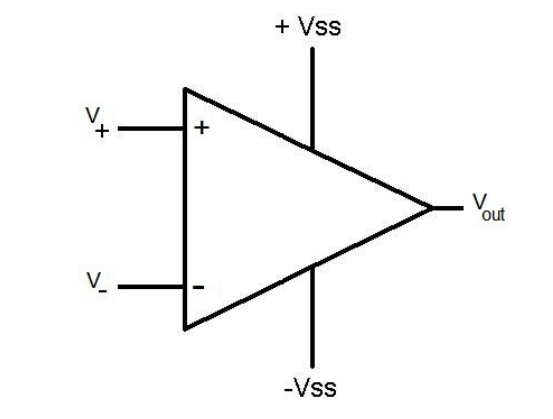
Grazie allo switch collegato alla massa è possibile selezionare il circuito da analizzare: se posizionato come in Figura 14 il circuito sarà un raddrizzatore a singola semionda, mentre commutando lo switch si otterrà un circuito raddrizzatore a doppia semionda. Come si può vedere in figura è presente un filtro per evitare che la VOUT sia attraversata da tensioni indesiderate. Tale filtro è composto da un condensatore da 1 micro Farad in serie ad una resistenza da 10 kOhm ed in parallelo con un diodo Zener da 4,7 volt.

4.4 Amplificatore Operazionale

L'amplificatore operazionale (Figura 15.) è un componente elettronico analogico, la cui funzionalità principale è l'amplificazione dei segnali al suo ingresso. Il nome di questo dispositivo è dovuto al suo largo impiego in elettronica per implementare operazioni matematiche sui segnali elettrici come addizioni, sottrazioni, ma anche operazioni complesse come ad esempio logaritmi, integrali e derivate.

Nel progetto si è presentata la necessità di inserire un amplificatore operazionale per amplificare la tensione all'interno dei circuiti. In Figura 15 il suo disegno circuitale e di seguito le configurazioni utilizzate:

Figura 15. Amplificatore Operazionale



Fonte: <http://www.robertopasini.com/index.php/2-uncategorised/495-hw-alimentazione-operazionale>

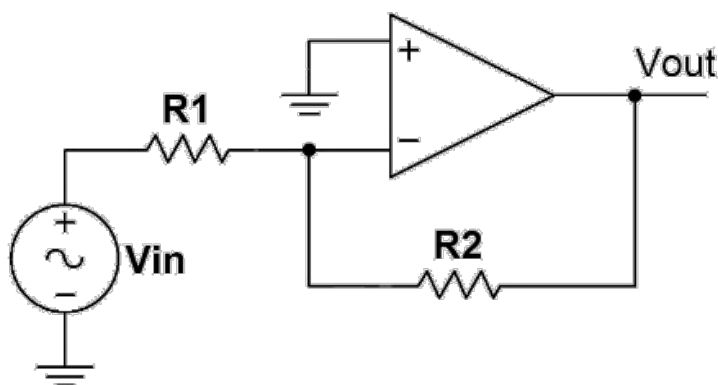
4.4.1 Amplificatore Operazionale Configurazione Invertente

L'amplificatore invertente (Figura 16.) è un circuito largamente utilizzato, in esso la tensione V_{in} è applicata all'ingresso invertente, mentre l'altro ingresso, detto non invertente, viene collegato a massa, cioè a una tensione di 0 V. Il segnale in uscita (Figura 17.) risulta sfasato di 180° , da qui il nome di amplificatore invertente. Il guadagno di questo circuito è: $G = -R_2/R_1$ per cui: $V_{out} = - (R_2/R_1)V_{in}$

Un amplificatore in configurazione invertente è costituito da due resistenze ed un amplificatore operazionale.

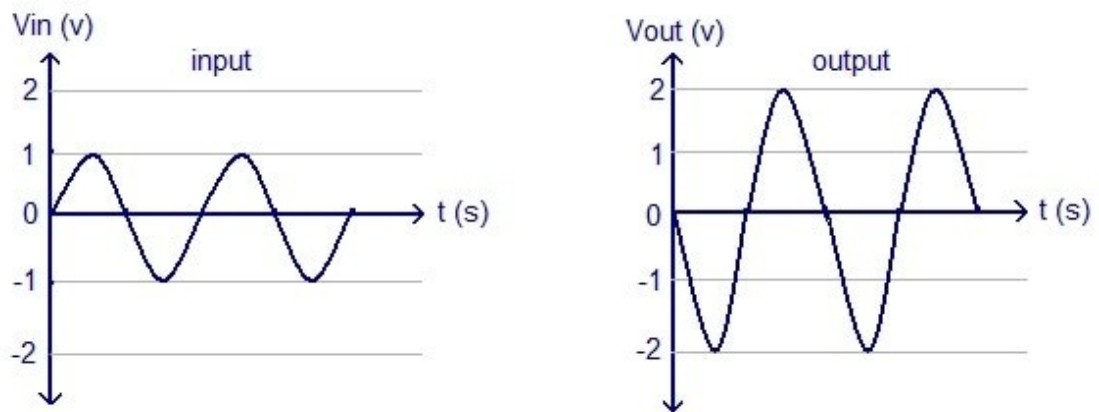
- R_2 connette l'uscita all'ingresso invertente. Si tratta quindi di un circuito a retroazione negativa
- La tensione di ingresso è collegata all'ingresso invertente dell'operazionale tramite R_1
- L'ingresso non invertente è connesso a massa

Figura 16. Amplificatore Operazionale configurazione Invertente



Fonte: <https://elettronicasemplice.weebly.com/amplificatore-operazionale-invertente.html>

Figura 17. Uscita Invertita e Amplificata



Fonte: <https://www.progettiarduino.com/amplificatore-operazionale-invertente-con-esercizi-e-simulazione.html>

4.4.2 Amplificatore Operazionale Configurazione non Invertente

Nel circuito in Figura 18. il guadagno dipende dal rapporto tra le resistenze, ma in questo caso il segnale in uscita è in fase con quello in ingresso.

In questo caso il guadagno è: $G = 1 + (R2/R1)$

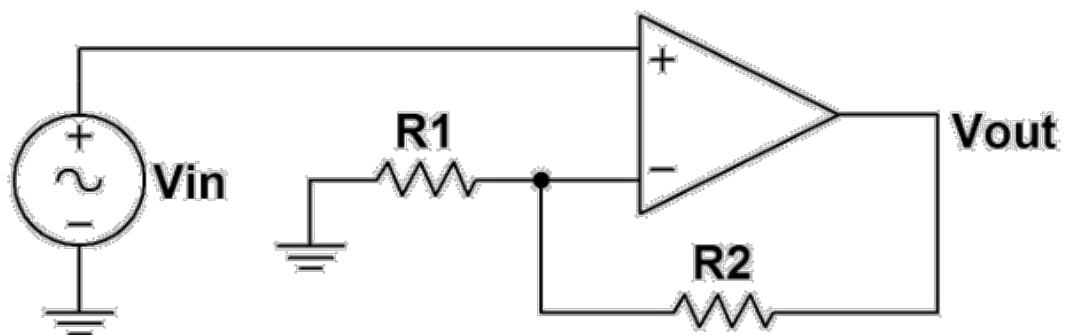
e la relazione finale è: $V_{out} = 1 + (R2/R1)V_{in}$

Un amplificatore in configurazione non invertente è costituito da due resistenze ed un amplificatore operazionale.

- R2 connette l'uscita all'ingresso invertente
- La tensione di ingresso è collegata direttamente all'ingresso non invertente dell'operazionale
- L'ingresso invertente è connesso a massa tramite R1

Se si confronta tale schema con quello dell'amplificatore invertente si nota che V_{in} e massa in ingresso sono collegati "al contrario".

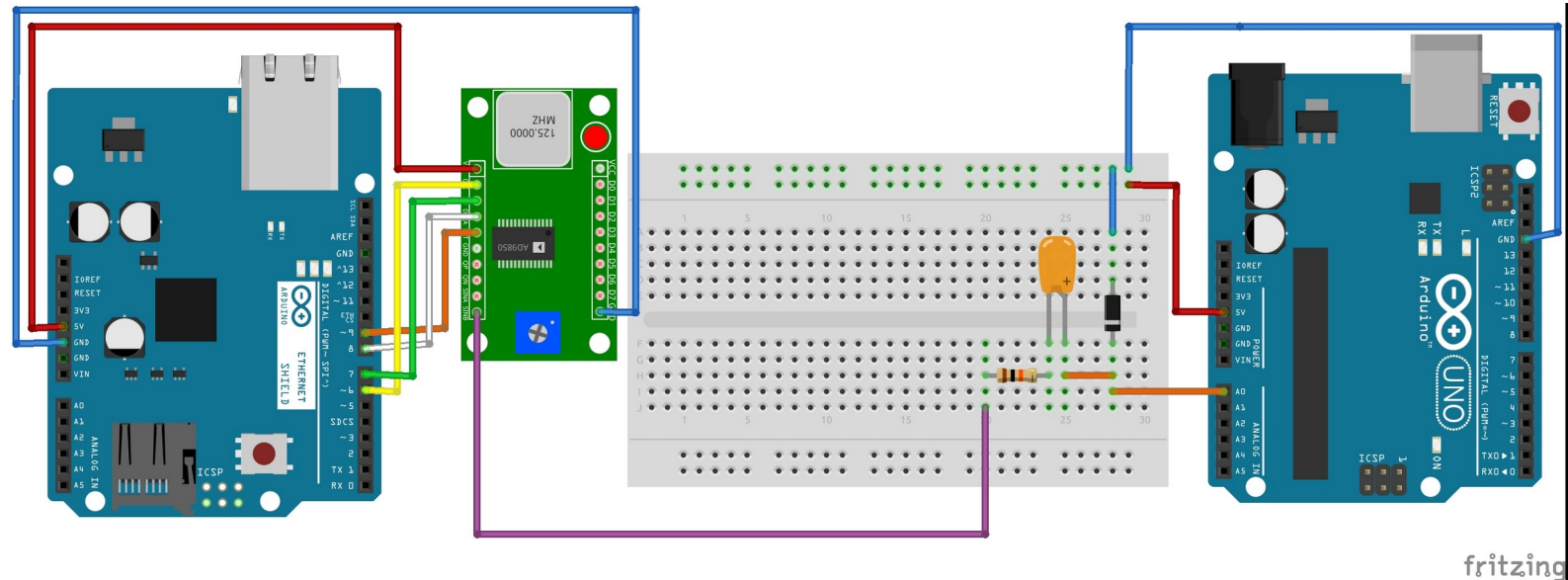
Figura 18. Amplificatore Operazionale configurazione non Invertente



Fonte: <https://elettronicasemplice.weebly.com/amplificatore-operazionale-non-invertente.html>

5 Oscilloscopio con Arduino

Figura 19. Circuito con AD9850, filtro e Arduino Oscilloscopio



Il circuito in Figura 19. è composto nella parte sinistra da Arduino collegato con l'integrato AD9850, spiegato nel capitolo 3, mentre la parte destra verrà analizzata nel seguente capitolo.

Come visto nel capitolo 4 si utilizza un filtro, composto da un resistore da 10KOhm posto in serie con un condensatore non polarizzato da 1 Micro Farad e in parallelo con un diodo Zener da 4,7 Volt, per non far attraversare il pin di ingresso di Arduino (A0) da tensioni indesiderate. Tale filtro può anche essere omesso considerato il basso voltaggio uscente dall'AD9850.

Gli ingressi analogici di Arduino sono in grado di leggere valori di tensione compresi nel range tra 0 e 5 Volt. Avendo Arduino un convertitore analogico digitale a 10 bit avrò 1024 valori di tensione. Dividendo i 5 volt per i 1024 valori restituiti dal convertitore analogico digitale si otterrà un valore unitario di 4,8 milli Volt.

Tramite il seguente Sketch Arduino lavora come un oscilloscopio:

```
#define Campioni 300 //campioni presi in memoria più sono e più preciso sarà il segnale
riportato
#define FrequenzaCampionamento 10000 //frequenza di campionamento arduino arriva
fino a 9/10k
unsigned int periodo;
unsigned long microsecondi;
double valore[Campioni];
```



```

void setup() {
    Serial.begin(1200); //inizializza la porta seriale a 1200 bound
    periodo = round(1000000*(1.0/FrequenzaCampionamento)); //trasforma frequenza di
    campionamento in periodo in microsecondi
}

void loop() {
    for (int i = 0; i < Campioni; i++) {
        microsecondi = micros();
        valore[i] = analogRead(A0); //setta la variabile "valore[i]" con il valore letto dalla porta
        analogica A0
        while(micros() < (microsecondi + periodo)); //ritarda il campionamento per un tempo
        pari al nostro periodo
    }

    for (int i = 0; i < Campioni; i++) {

        Serial.println(map((valore[i]),0,1024,0,500)); //stampa tramite la seriale il valore i-esimo
        della variabile "valore[i]". La funzione map trasforma il valore preso da 0 a 500 anzichè da
        0 a 1024
    }
    delay(100); //ritardo in millisecondi

    //con while(1); si ottiene un singolo campionamento altrimenti si utilizza un delay di
    qualche milli secondo

}

```

5.1 Analizzatore di spettro con Arduino

Analogamente a quanto visto precedentemente se invece di studiare l'andamento del segnale rispetto al dominio del tempo si vuole avere la densità spettrale del segnale, quindi studiare il segnale nel dominio della frequenza, si ha bisogno di un analizzatore di spettro. Per trasformare un segnale dal dominio del tempo al dominio della frequenza si utilizza la trasformata di Fourier. Arduino implementa delle librerie che eseguono la FFT (Fast Fourier Transform) grazie alle quali esso può assumere la funzione di analizzatore di spettro.

Eseguendo gli stessi collegamenti visti in Figura 19. tramite il seguente Sketch si può ottenere un analizzatore di spettro:

```
#include "arduinoFFT.h" //libreria FFT Fast Fourier Transform

#define Campioni 128 //numero di campioni presi
#define FrequenzaCampionamento 10000 //la frequenza di campionamento di arduino
arriva fino a 9/10k

arduinoFFT FFT = arduinoFFT(); //crea una variabile di tipo "arduinoFFT"

unsigned int periodo;
unsigned long microsecondi;

double valReale[Campioni]; //variabile che assumerà i valori reali
double valImag[Campioni]; //variabile che assumerà i valori immaginari

void setup() {
    Serial.begin(115200); //inizializza la porta seriale a 115200 bound
    periodo = round(1000000*(1.0/FrequenzaCampionamento)); //trasforma frequenza di
    campionamento in periodo in microsec
}

void loop() {
    for (int i = 0; i < Campioni; i++) {
        microsecondi = micros();
        valReale[i] = analogRead(A0); //la variabile "valReale[i]" acquisisce il valore letto dal
        pin analogico A0
        valImag[i] = 0;
```

```

        while(micros() < (microsecondi + periodo)){}; //ritarda il campionamento per un
tempo pari al nostro periodo
    }

    FFT.Windowing(valReale,    Campioni,    FFT_WIN_TYP_HAMMING,
FFT_FORWARD);// trasforma i valori temporali in valori di frequenza

    FFT.Compute(valReale, valImag, Campioni, FFT_FORWARD); //il metodo precedente li
prepara e questo li calcola. Produce numeri complessi (spazi a 2 coordinate)

    FFT.ComplexToMagnitude(valReale,  valImag,  Campioni); //per problemi di
rappresentazione dei numeri complessi questo metodo ne calcola la potenza

    // se si vuole trovare la frequenza di picco nello spettro: double peak =
FFT.MajorPeak(valReale, Campioni, FrequenzaCampionamento);

    //Serial.println(peak);

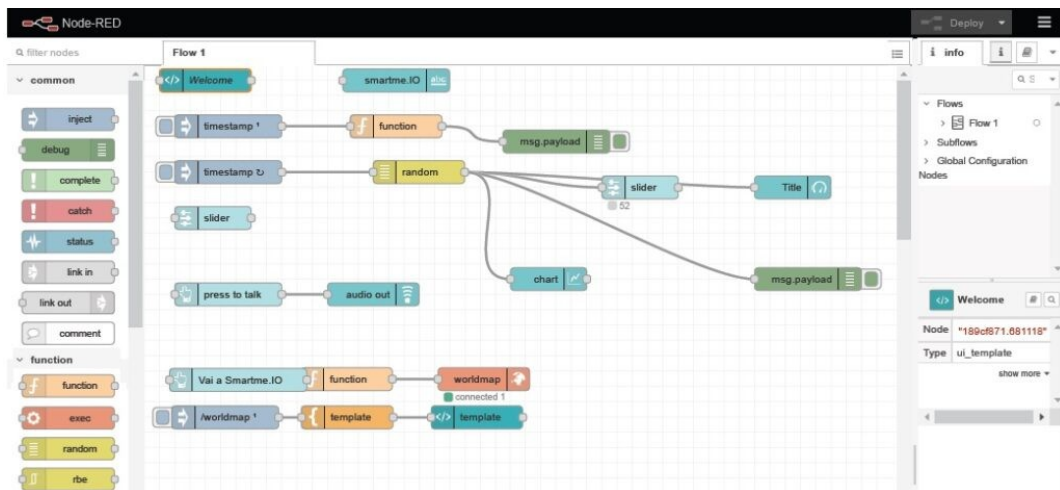
    for (int i = 0; i < Campioni/2; i++) {
        Serial.println(valReale[i],1);
    }
    //delay(1000);
    while(1); //per fare un campionamento altrimenti si utilizza un delay di qualche secondo
}

```

6 Node-Red

Node-RED è un linguaggio di programmazione a blocchi, sviluppato per gestire agevolmente flussi di dati che viaggiano in ambiente IoT , o generati dai sensori gestiti da baord attraverso la programmazione flow-based. Node-red mette a disposizione degli utenti un editor di flussi (Figura 20.) basato su browser, che grazie ad un'ampia galleria di nodi presenti nella palette della dashboard, semplifica il collegamento e la realizzazione dei flussi, che possono essere distribuiti in runtime. Le funzioni JavaScript possono essere create all'interno dell'editor utilizzando un editor di testo avanzato. Le funzioni, i modelli o i flussi creati dall'utente, grazie ad una libreria integrata, possono essere salvati per il riutilizzo.

Figura 20. Node-Red



Fonte: <https://arancino.cc/docs/software/node-red/>

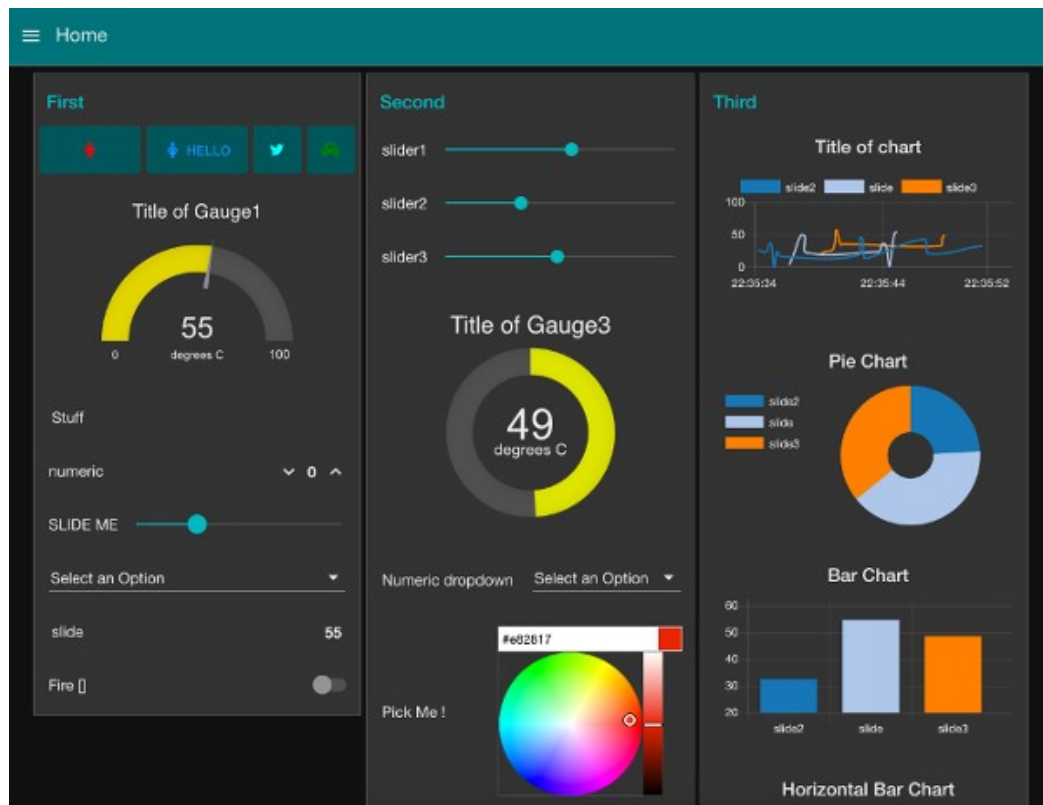
Node-red si basa sul protocollo MQTT (Message Queuing Telemetry Transport) di IBM, lo standard di riferimento della comunicazione per l'Internet delle Cose. Le informazioni tra i nodi di ogni flusso viaggiano sotto forma di messaggi (msg) alfa-numerici o di altro tipo (json), contenuti nei blocchi payload o msg.payload ; ogni messaggio inviato ad un nodo del flusso, può essere inviato al nodo successivo, che lo userà a sua volta per l'operazione per cui è stato programmato.

Il runtime è basato su Node.js, sfruttando appieno il suo modello non bloccante basato sugli eventi. Questo lo rende ideale per funzionare ai margini della rete su hardware a basso costo e nel Cloud. I flussi creati in Node-RED vengono archiviati utilizzando JSON, tipo di file che può essere facilmente importato ed esportato per la condivisione. Una libreria di flussi online consente di condividere i flussi realizzati.

6.1 Dashboard di Node-Red

Node Red può implementare diverse funzioni una di queste è la Dashboard (Figura 21.). Questo modulo fornisce un set di nodi in Node-RED per creare rapidamente una dashboard di dati live.

Figura 21. Dashboard di Node-Red



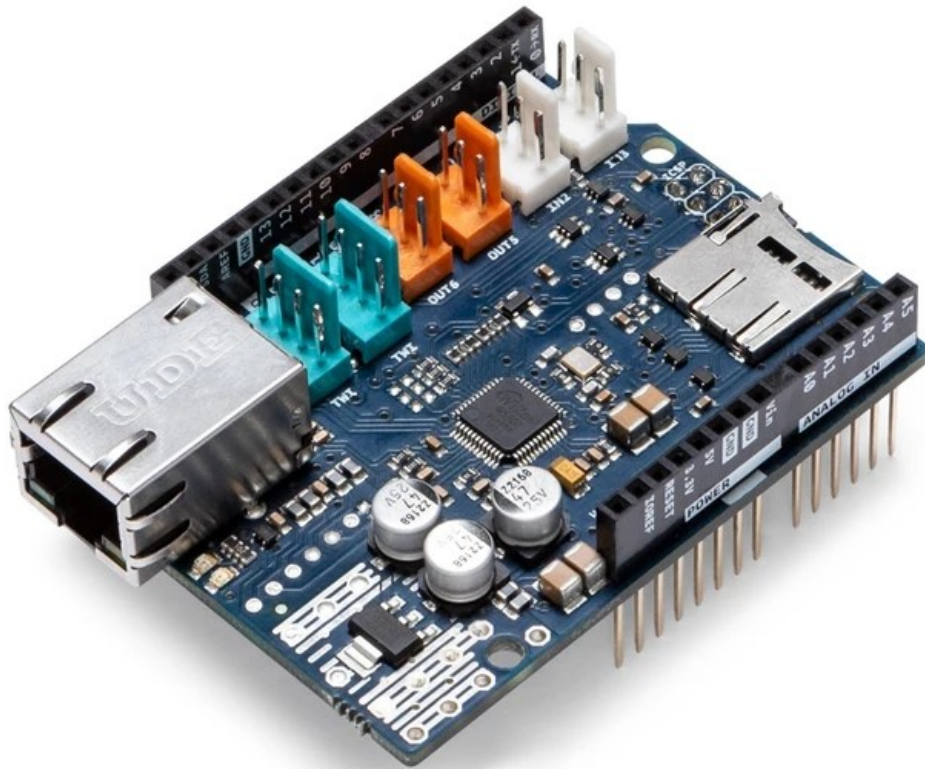
Fonte: <https://flows.nodered.org/node/node-red-dashboard>

In questo progetto la Dashboard verrà utilizzata come strumento per creare un grafico attraverso i dati letti via Seriale provenienti da Arduino.

7 Arduino Ethernet Shield

L' Arduino Ethernet Shield (Figura 22.) consente ad una scheda Arduino di connettersi a internet utilizzando la libreria Ethernet e di leggere e scrivere su una scheda SD utilizzando la libreria SD.

Figura 22. Arduino Ethernet Shield



Fonte: <https://www.arduino.cc/en/Guide/ArduinoEthernetShield>

Si collega al pc o a un hub di rete o router utilizzando un cavo ethernet standard (CAT5 o CAT6 con connettori RJ45). La connessione a un computer potrebbe richiedere l'uso di un cavo incrociato (sebbene molti computer possano eseguire il crossover internamente).

Di seguito tutte le funzionalità che offre la libreria Ethernet:

- ChatServer : imposta un semplice server di chat.
- WebClient : effettua una richiesta HTTP.
- WebClientRepeating : effettua richieste HTTP ripetute.

- WebServer : ospita una semplice pagina HTML che visualizza i valori dei sensori analogici.
- BarometricPressureWebServer : emette i valori da un sensore di pressione barometrica
- UDPSendReceiveString : invia e ricevi stringhe di testo tramite UDP.
- UdpNtpClient : interroga un server Network Time Protocol (NTP) utilizzando UDP.
- DnsWebClient : client Web basato su DNS e DHCP.
- DhcpChatServer : un semplice server di chat DHCP
- DhcpAddressPrinter : Ottiene un indirizzo IP tramite DHCP e lo stampa
- TelnetClient : un client Telnet

Allo shield deve essere assegnato un indirizzo MAC e un indirizzo IP fisso utilizzando la funzione `Ethernet.begin()`. Un indirizzo MAC è un identificatore univoco globale per un particolare dispositivo. È possibile utilizzare DHCP per assegnare dinamicamente un IP allo shield. Facoltativamente, si può anche specificare un gateway di rete e una sottorete.

7.1 Arduino Web Server

Appurato che Arduino possa essere utilizzato come Web Server tramite l'utilizzo dell'Ethernet Shield, in questo progetto sarà Arduino a gestire il Server dove si svilupperà il sito.

Di seguito la bozza del codice per inizializzare un semplice sito web:

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

```
#include <Ethernet.h>
```

```
byte mac[]={0x04,0xD4,0xC4,0x15,0xB1,0x2C}; //indirizzo mac della scheda
```

```
byte ip[]={192,168,1,177}; //indirizzo IP della scheda
```

```
byte subnet[]={255,255,255,0}; //subnet scheda
```

```
byte gateway[]={192,168,1,1}; //gateway
```

```
EthernetServer server(80); //crea una variabile di tipo EthernetServer settando la porta 80 (servizi web "http")
```

```
void setup() {
```

```
    Ethernet.begin(mac,ip,gateway,subnet); //inizializza la scheda Ethernet con i valori impostati precedentemente
```

```

}

void loop() {
  EthernetClient client =server.available(); //setta la variabile client come server attivo
  if (client) { //se il client esiste..
    while(client.connected()) { //finche il client è connesso alla rete
      if(client.available()){ //se il client è attivo

        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.println("<html>");
        client.println("<head><title>Accendi/Spegni led</title>");
        client.println("<link rel='shortcut icon' type='image/x-icon'
href='http://i44.servimg.com/u/f44/16/84/89/65/23570310.png' /link>");
        client.println("</head>");
        client.println("<body><a Scrivi quello che vuoi /a></body>");
        client.println("</html>"); //fine codice html
        client.stop(); //stop richieste dal client
      }
    }
  }
}

```


8 IL PROGETTO

Attraverso Arduino (Capitolo 1), le sue molteplici implementazioni (Capitolo 3,5,7), Raspberry Pi (Capitolo 2) e i vari circuiti elettronici (Capitolo 4), si hanno le basi per poter progettare un laboratorio comandato da remoto.

Materiale necessario:

- 2 Schede Arduino
- Raspberry Pi (preferibilmente un modello superiore al 3)
- Pi Camera
- AD9850
- Ethernet Shield Arduino
- Cavo Ethernet RJ45 incrociato
- Breadboard o millefori
- LM324N (integrato composto da 4 amplificatori operazionali)
- Relè impulsivo a 5V
- 4 diodi raddrizzatori
- 9 resistenze da 10kilo Ohm
- Condensatore (di qualsiasi capacità in questo progetto si è utilizzato un condensatore non polarizzato da 1 micro Farad)

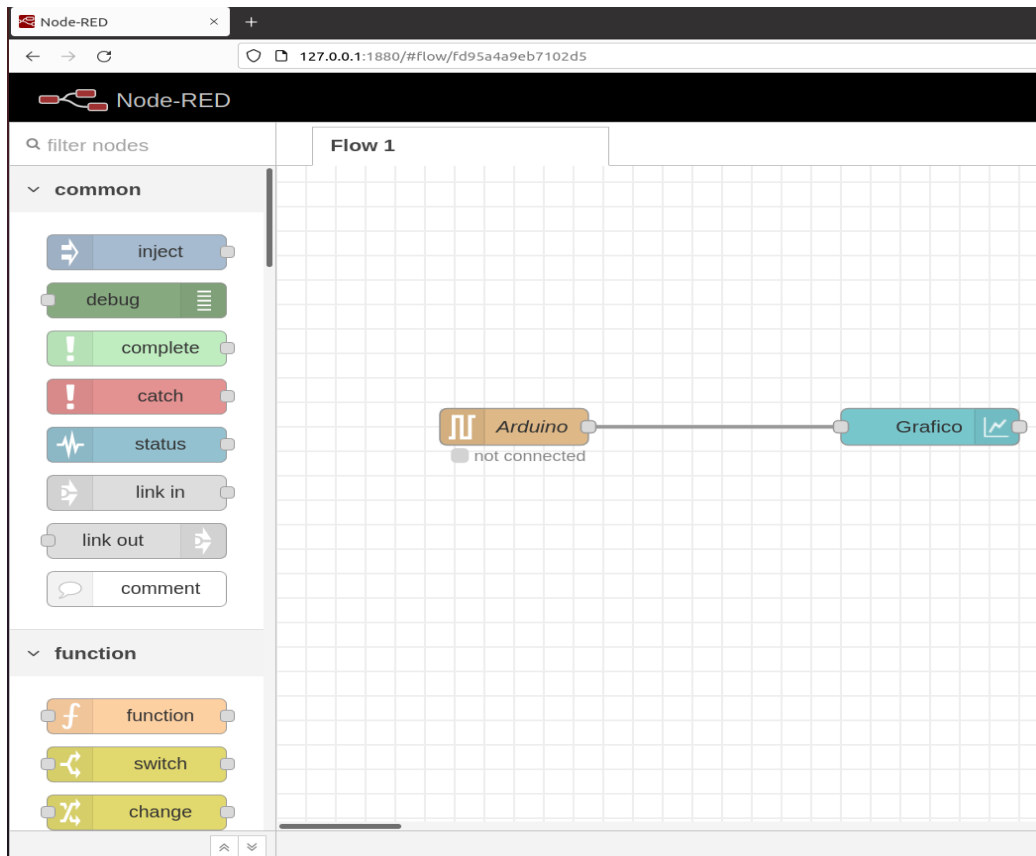
Installazione:

Nel Raspberry Pi, dotato di sistema operativo Raspian, si installa Node.js, npm e Node Red.

Successivamente si installano i nodi della dashboard e delle porte seriali tramite il comando npm.

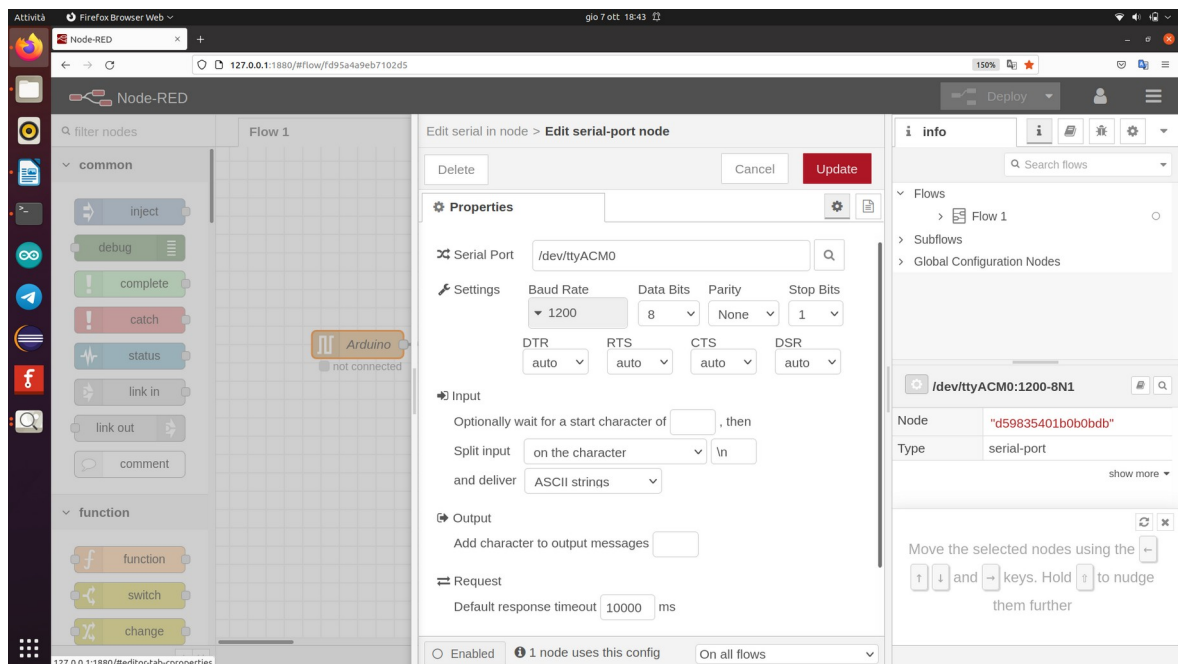
Si accede a Node Red e si esegue lo schema in Figura 23:

Figura 23. Diagramma Node-Red



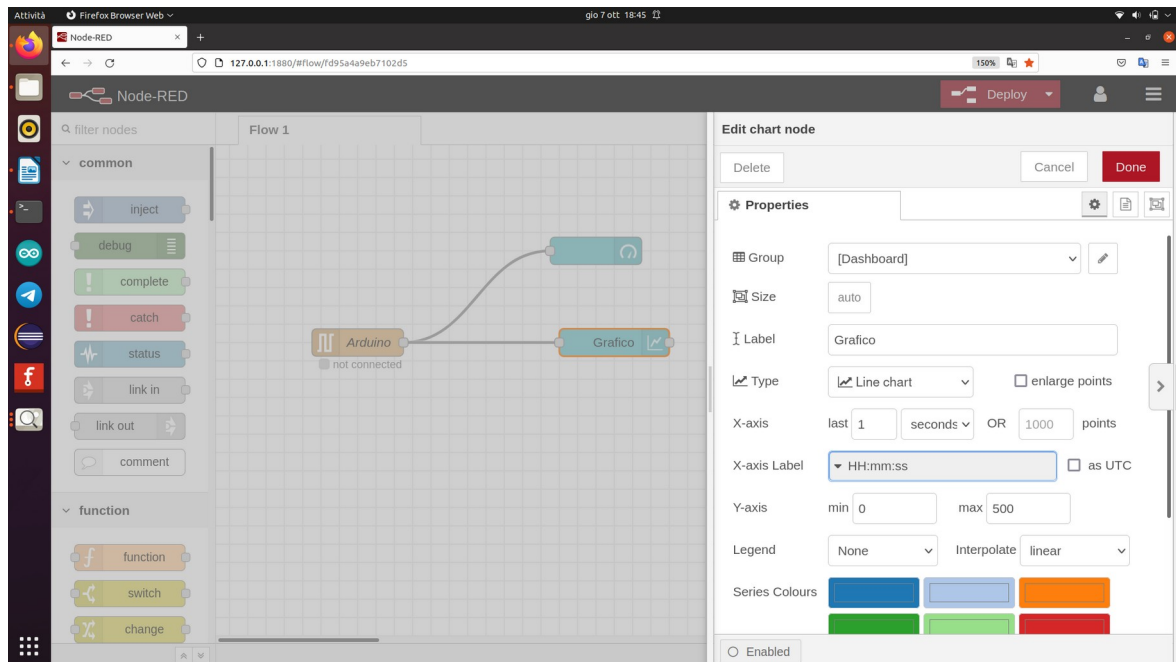
La porta seriale è configurata come in figura 24:

Figura 24. Configurazione Porta Seriale



Mentre la configurazione del grafico è in Figura 25:

Figura 25. Configurazione Grafico Node-Red



Node-Red può essere avviato in automatico all'accensione del Raspberry tramite la seguente riga di comando inserita nel terminale:

```
sudo systemctl enable nodered.service
```

Una volta inizializzato e configurato Node-Red si esegue la procedura presente nel Capitolo 2.2.1 per connettere la Pi Camera alla rete.

Successivamente si installa l'IDE di Arduino dal sito ufficiale.

Si installa fisicamente l'Ethernet Shield su Arduino e si collega l'AD9850 come spiegato nel Capitolo 3.

Arduino va implementato con il seguente codice all'interno dello Sketch per essere utilizzato come un Web Server e acquisisca da remoto le istruzioni per comandare il generatore di segnali AD9850 :

```
#include <AD9850.h>
```

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

```
#include <Ethernet.h>
```

```
#define rele 4
```

```

byte mac[]={0x04,0xD4,0xC4,0x15,0xB1,0x2C}; //indirizzo MAC
byte ip[]= {192,168,1,177}; //indirizzo IP a scelta
byte subnet[]={255,255,255,0}; // indirizzo subnet
byte gateway[]={192,168,1,1}; // indirizzo gateway
EthernetServer server(80); //variabile server settato con la porta 80
String ascoltatore=""; //variabile che acquisirà le stringhe provenienti dall'URL
double freq = 10.0; // variabile dove verrà impostato il valore della frequenza
int phase = 0;

void setup() {
  Ethernet.begin(mac,ip,gateway,subnet); //setta l'eternet con i parametri scelti precedentemente
  DDS.begin(6, 7, 8, 9); //setta i pin 6,7,8,9 di Arduino con l'AD9850
  DDS.calibrate ( 124999500 ); //valore di calibrazione dell'AD9850
  DDS.up(); //attiva l'AD9850
  DDS.setfreq(freq, phase); //comanda l'uscita dell'AD9850 con i valori di frequenza e fase impostati
  pinMode(rele,OUTPUT); //setta il relè (pin 4) come pin di uscita
}

void loop() {
  EthernetClient client =server.available();
  if (client) {
    while(client.connected()) {
      if(client.available()){ //istruzioni che garantiscono un corretto funzionamento del server
        char c=client.read(); //variabile "c" settata come carattere di lettura del URL
        ascoltatore.concat(c); //aggiunge alla stringa "ascoltatore" il carattere "c"
        if (c=="\n") { //se si è premuto invio

          if(ascoltatore.indexOf("wire=1")>0){ //se nella stringa "ascoltatore" è contenuta la parola tra parentesi
            digitalWrite(rele,LOW); //imposta il relè allo stato NC (normalmente chiuso)
          }
        }
      }
    }
  }
}

```

```

        if(ascoltatore.indexOf("wire=2")>0){ //se nella stringa "ascoltatore" è contenuta la
parola tra parentesi
            digitalWrite(rele,HIGH); //imposta il relè allo stato NO (normalmente aperto)
        }

        if(ascoltatore.indexOf("slot")>0){ //se nella stringa "ascoltatore" è contenuta la
parola tra parentesi
            int n=ascoltatore.indexOf("H"); //setta la variabile "n" come indice dove è contenuta
la lettera tra parentesi
            ascoltatore=ascoltatore.substring(11,n); //il contenuto della stringa "ascoltatore"
viene ridotto
            // a ciò che è compreso tra i caratteri posizionati dall'indice numero 11 all'indice
numero "n"
            freq=ascoltatore.toInt(); //il valore della frequenza viene settato attraverso la stringa
"ascoltatore" sottoforma di numero intero
            DDS.setfreq(freq, phase); //comanda l'uscita dell'AD9850 con i valori di frequenza e
fase impostati

            delay(100); //ritardo di 100 millisecondi
        }

        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.println("<html>");
        client.println("<head><title>Laboratorio Virtuale</title> <link rel='shortcut icon'
type='image/x-icon' href='http://i44.servimg.com/u/f44/16/84/89/65/23570310.png'
/></head>");

        client.println("<body><table width='25%' height='10%'> <td><h2> Seleziona il circuito
</h2></td>");
            client.println("<tr><td><a href='/?wire=1'><button><h2>Singola
Semionda</h2></button></a> - <a href='/?wire=2'><button><h2>Doppia
Semionda</h2></button></a></td></table>");

        client.println("<h3> Inserisci la Frequenza</h3>");

```

```
client.println("<script> function fetch() {var get=document.getElementById('get').value;  
document.getElementById('put').value=get; }</script> ");
```

```
client.println("<label for='quantity'><h4>Seleziona una Frequenza dalla barra (1Hz -  
2000Hz): </h4></label>");
```

```
client.println("<form method='get'> <input name='slot' type='range' min='1' max='2000'  
id='get'  
onchange='fetch()'style=width:28%;height:10%;/><br>"); //style=width:550px;height:100  
px;'
```

```
client.println("<input type='number' id='put'> <input type='submit'  
value='invia'></form>");
```

```
client.println("<object data='http://progettotesi.ddns.net:1880/ui/' width='40%'  
height='60%'></object>");
```

```
client.println("<object data='http://IP_Raspberry:1880/ui/' width='40%'  
height='60%'></object>");
```

```
client.println("<object data='http://progettotesi.ddns.net:8000' width='50%'  
height='80%'align='right'></object>");
```

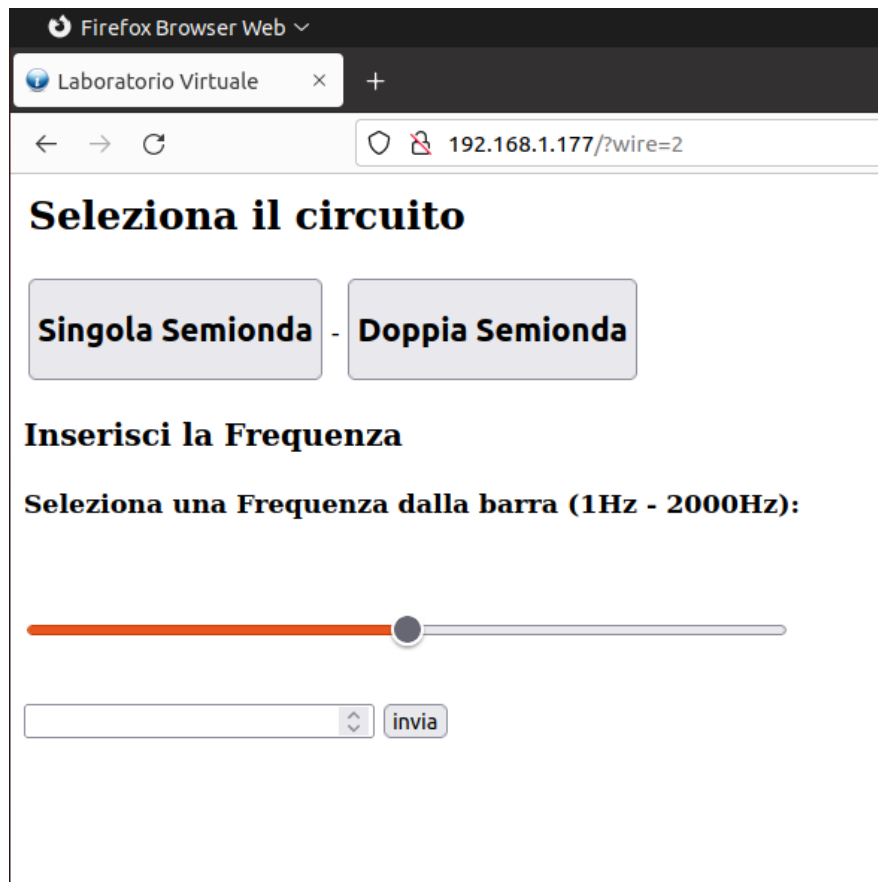
```
client.println("<object data='http://IP_Raspberry:8000' width='50%'  
height='80%'align='right'></object></body></html>");
```

```
client.stop(); //stop richieste dal client  
ascoltatore=""; //reseta la stringa "ascoltatore"  
}  
}  
}  
}  
}
```

Ciò che è contenuto all'interno delle istruzioni "client.println();" è codice in linguaggio HTML, CSS e JavaScript e serve per implementare il sito Web.

Alimentando Arduino Web Server e collegandolo al router con un cavo Ethernet Incrociato si può accedere localmente inserendo nel browser l'indirizzo 192.168.1.177 (indirizzo dato ad Arduino Web Server dallo Sketch) visualizzando la pagina in figura 26.

Figura 26. Pagina Iniziale



Per visualizzarla da remoto è necessario aprire le porte del router che si sono utilizzate per Arduino Web Server, Node-Red e lo streaming Web della Pi Camera.

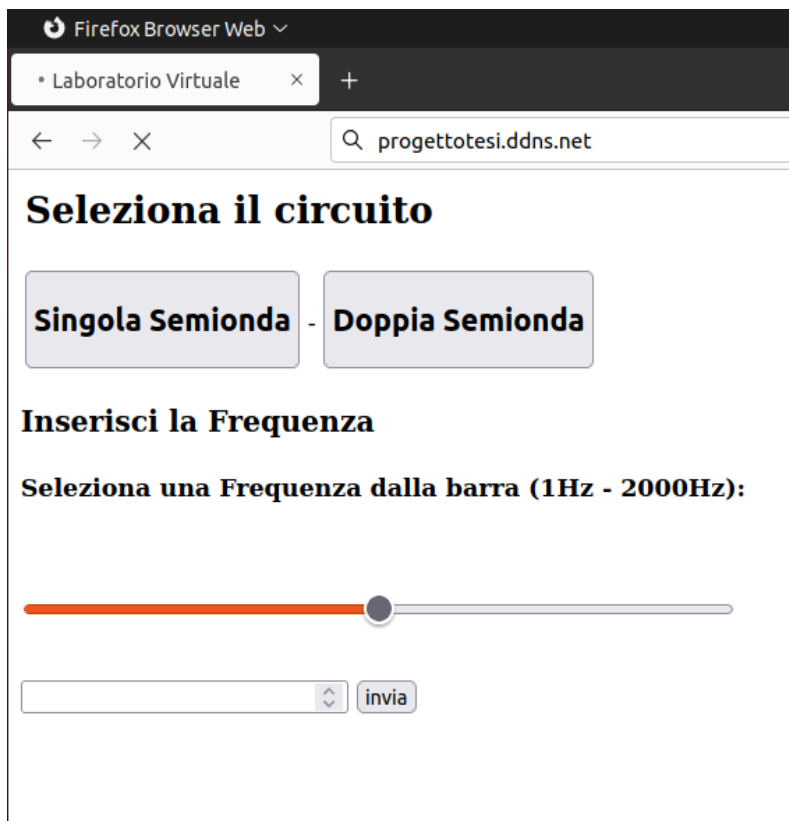
Nello sviluppo in esame sono utilizzate le seguenti porte:

- 80 Arduino Web Server
- 1880 Node-Red
- 8000 Pi Camera Web

Una volta eseguito quest'ultimo passaggio bisogna conoscere il proprio IP dinamico dato dall'Internet Service Provider ed inserire quest'ultimo per connettersi da remoto.

Per utilizzare un DDNS (Dynamic Domain Name Server) è necessario richiedere il dominio ad un sito che li fornisce. In questo progetto si è utilizzato un DDNS denominato "progettotesi.ddns.net" realizzato tramite il sito NO-IP.com (Figura 27).

Figura 27. DDNS



Firefox Browser Web

Laboratorio Virtuale

progettotesi.ddns.net

Seleziona il circuito

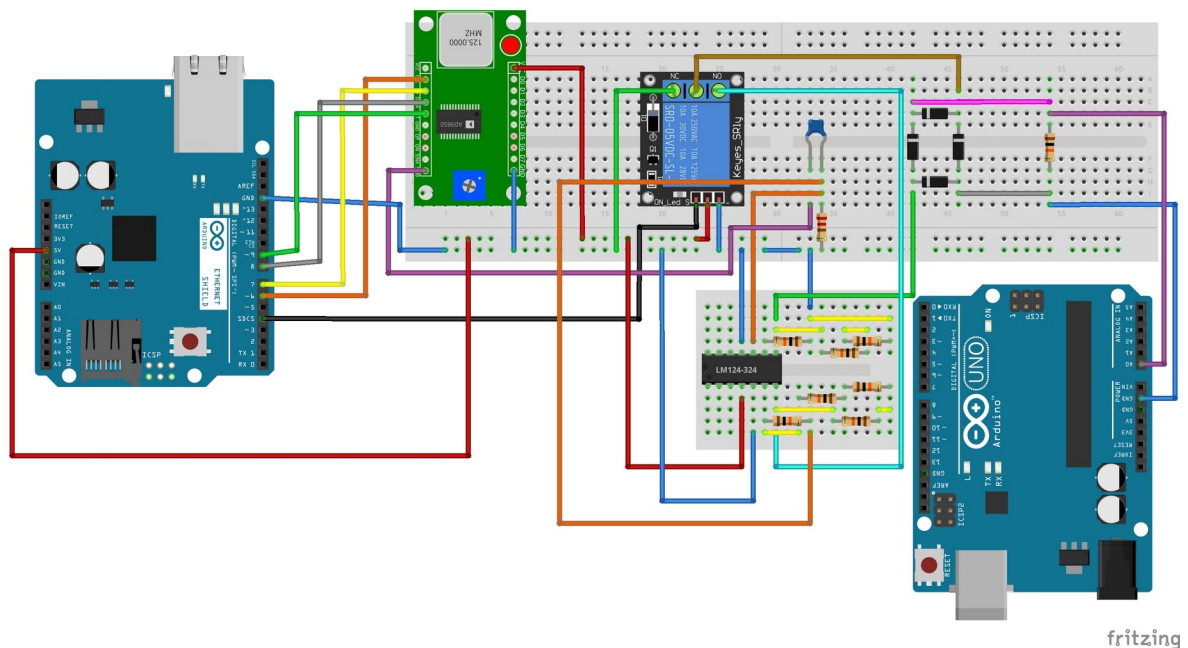
Singola Semionda - Doppia Semionda

Inserisci la Frequenza

Seleziona una Frequenza dalla barra (1Hz - 2000Hz):

Una volta terminata la fase delle configurazioni Web si prosegue cablando l'uscita dell'AD9850 all'ingresso del circuito da analizzare spiegato nel capito 4.3 e l'uscita del circuito, all'ingresso dell'Arduino utilizzato come oscilloscopio (Figura 28.).

Figura 28. Cablaggio Progetto



Il filtro all'uscita dell'AD9850, composto da un condensatore con una resistenza, è un filtro passa alto che ha la funzione di azzerare il valor medio che in uscita dall'AD9850 è di 600 mV.

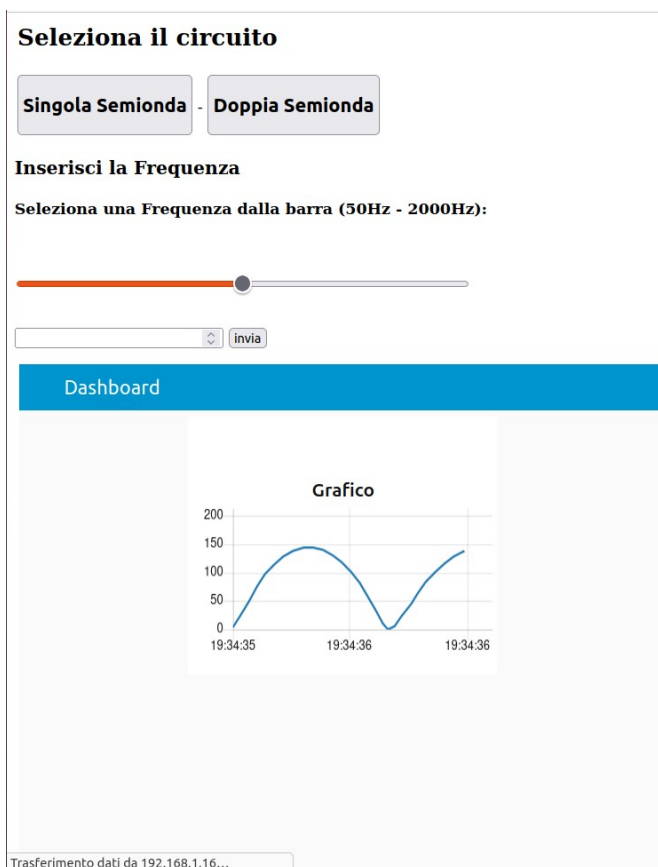
All'uscita del filtro abbiamo quindi un segnale sinusoidale con valor medio nullo e ampiezza 1,2V (da 0,6V a -0,6V). I diodi hanno un assorbimento di 0,7V l'uno. Il segnale uscente dal raddrizzatore sarebbe nullo, quindi per amplificare il segnale si utilizza un integrato LM320N avente quattro amplificatori operazionali.

Nel circuito in Figura 28 si può notare che dei quattro amplificatori ne sono utilizzati solamente due, uno in configurazione non invertente e l'altro in configurazione invertente. Dall'uscita del amplificatore invertente si ottiene una tensione: $V_{out}=(-3R:R)V_{in}$; ciò significa che si avrà il valore della tensione di ingresso invertito (sfasato di 180°) e moltiplicato per tre. Mentre all'uscita del amplificatore non invertente si ottiene una tensione: $V_{out}=(1+2R:R)V_{in}$; ciò significa che si avrà il valore della tensione di ingresso moltiplicato per tre. Ottenendo un segnale sinusoidale con ampiezza 3,6V (da 1,8V a -1,8V) si può ottenere l'uscita attesa.

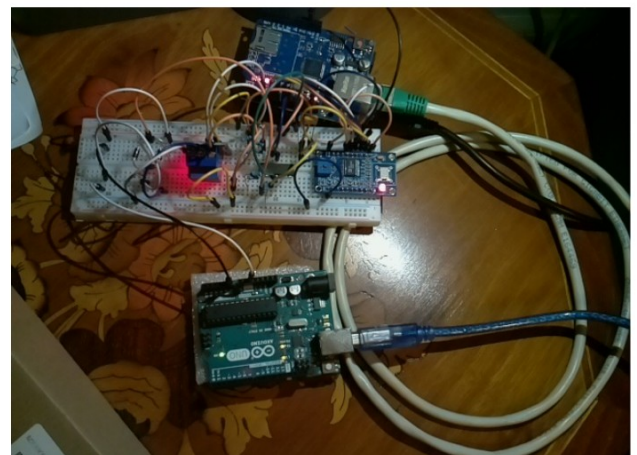
Il filtro finale che è presente nel capitolo 4.3, composto dal condensatore in serie con la resistenza ed in parallelo con il diodo Zener, è stato omesso considerata la bassa tensione uscente dal circuito.

Collegando tutto al Raspberry si può ottenere quanto segue dall'interfaccia Web (Figura 29).

Figura 29. Interfaccia Web



PiCamera Progetto Streaming



Conclusioni e Sviluppi futuri

Il laboratorio remoto presentato, può essere una soluzione innovativa per una futura didattica e/o progettazione lavorativa a distanza. Nel progetto sono stati utilizzati componenti elettronici a basso costo e quindi con un livello di precisione approssimativo, ma efficiente per lo scopo del progetto.

Mettendo a paragone l'hardware dei componenti utilizzati, Raspberry e Arduino possono essere sostituiti con dei PLC (*programmable logic controller*) che offrono prestazioni migliori anche se con costi più elevati.

A livello software, il server potrebbe essere gestito da Raspberry invece che da Arduino, ottenendo la possibilità di aggiungere più codice CSS e HTML. La limitazione di Arduino UNO in questo progetto è quella di sfruttare la memoria dinamica oltre il 70% (spingerla oltre il 75-80% significherebbe creare problematiche alla stabilità del sistema).

Altra soluzione sarebbe stata quella di utilizzare Node-Red, non solo per le funzioni che offre la Dashboard; ma anche per mettere in comunicazione, tramite il protocollo MQTT (Message Queue Telemetry Transport), il sito Web, Raspberry, Arduino con AD9850 e Arduino Oscilloscopio.

Il laboratorio realizzato è un progetto di base che potrebbe essere in futuro sviluppato con l'utilizzo di risorse tecniche ed economiche maggiori. In questo modo, si avrebbe la possibilità di sviluppare un prodotto aziendale con un vasto numero di funzionalità ed una migliore efficienza tecnica; l'utente potrà utilizzare questo laboratorio da remoto senza percepirne la differenza con uno reale.

• **Bibliografia**

Biondo Giuseppe - Sacchi Enrico, “*Manuale di elettronica e telecomunicazioni*”, HOEPLI, 2015.

Michael Margolis, “*Arduino. Progetti e soluzioni*”, Tecniche Nuove, 2013.

Massimo Banzi - Michael Shiloh, “*Arduino La guida ufficiale*”, Tecniche Nuove, 2015.

Paolo Aliverti, “*Elettronica per maker*”, Edizioni LSWR, 2015.

Paolo Aliverti, “*Il manuale di Arduino*”, Edizioni LSWR, 2016.

Paolo Aliverti, “*Arduino. Trucchi e segreti. 120 idee per risolvere ogni problema*”, Edizioni LSWR, 2018.

• **Sitografia**

Sito Ufficiale di Arduino:

<https://store.arduino.cc/products/arduino-uno-rev3/>

<https://www.arduino.cc/en/Guide/ArduinoEthernetShield>

Sito Ufficiale di Raspberry Pi:

<https://www.raspberrypi.org/documentation/accessories/camera.html>

Sito Ufficiale della Pi Camera:

<https://picamera.readthedocs.io/en/release-1.13/recipes2.html#web-streaming>

Sito Ufficiale Node-Red:

<https://flows.nodered.org/node/node-red-dashboard>

Altri link:

[https://it.wikipedia.org/wiki/Arduino_\(hardware\)](https://it.wikipedia.org/wiki/Arduino_(hardware))

https://it.wikipedia.org/wiki/Arduino_IDE

<https://support.microsoft.com/it-it/topic/caricamento-del-codice-bacheca-e-dell-ide-di-arduino-a9723765-1314-49e0-a69b-bb5c3e1f628d>

<http://pietrolodi.altervista.org/arduino9/>

<https://www.electronics-lab.com/project/using-arduino-ides-serial-plotter-feature/>

<https://www.amazon.it/Raspberry-Modello-Piastra-base-verde/dp/B07BFH96M3>

<https://www.mouser.it/new/raspberry-pi/raspberry-pi-3-bplus/>

<https://lorenzocasaburo.it/wp-content/uploads/2018/05/Copertina-Pi-Camera.jpg>

<https://opensource4u.com/lib/ad9850spi>

<http://www.elemania.altervista.org/diodi/pn/pn5.html>

<http://www.elemania.altervista.org/diodi/pn/pn6.html>

<http://www.robertopasini.com/index.php/2-uncategorised/495-hw-alimentazione-operazionale>

<https://elettronicasemplice.weebly.com/amplificatore-operazionale-invertente.html>

<https://www.progettiarduino.com/amplificatore-operazionale-invertente-con-esercizi-e-simulazione.html>

<https://elettronicasemplice.weebly.com/amplificatore-operazionale-non-invertente.html>

<https://arancino.cc/docs/software/node-red/>