I.T.I. OMAR

ESAME DI STATO
ANNO SCOLASTICO 2020-2021

ROBOCAP

PADOAN ANDREA
CLASSE 5'RA

Sommario

Introduzione:

The RoboCap project consists of 2 autonomous component parts created in two different moments, the MagicCap and the Pan Tilt. The MagicCap consists of a hat that we have specially developed for one of our disabled ex-partners. Part of his disability was that he was unable to move his arms, so in order to help him we decided to patent this cap. The operation consists in being able to move the mouse through the movement of the head and also to use the right and left click through a button that is held between the teeth. The second component is the PanTilt which emulates a traditional robotic arm. We have modified the sketch specifically to be able to use it with the MagicCap.

Elementi Base:

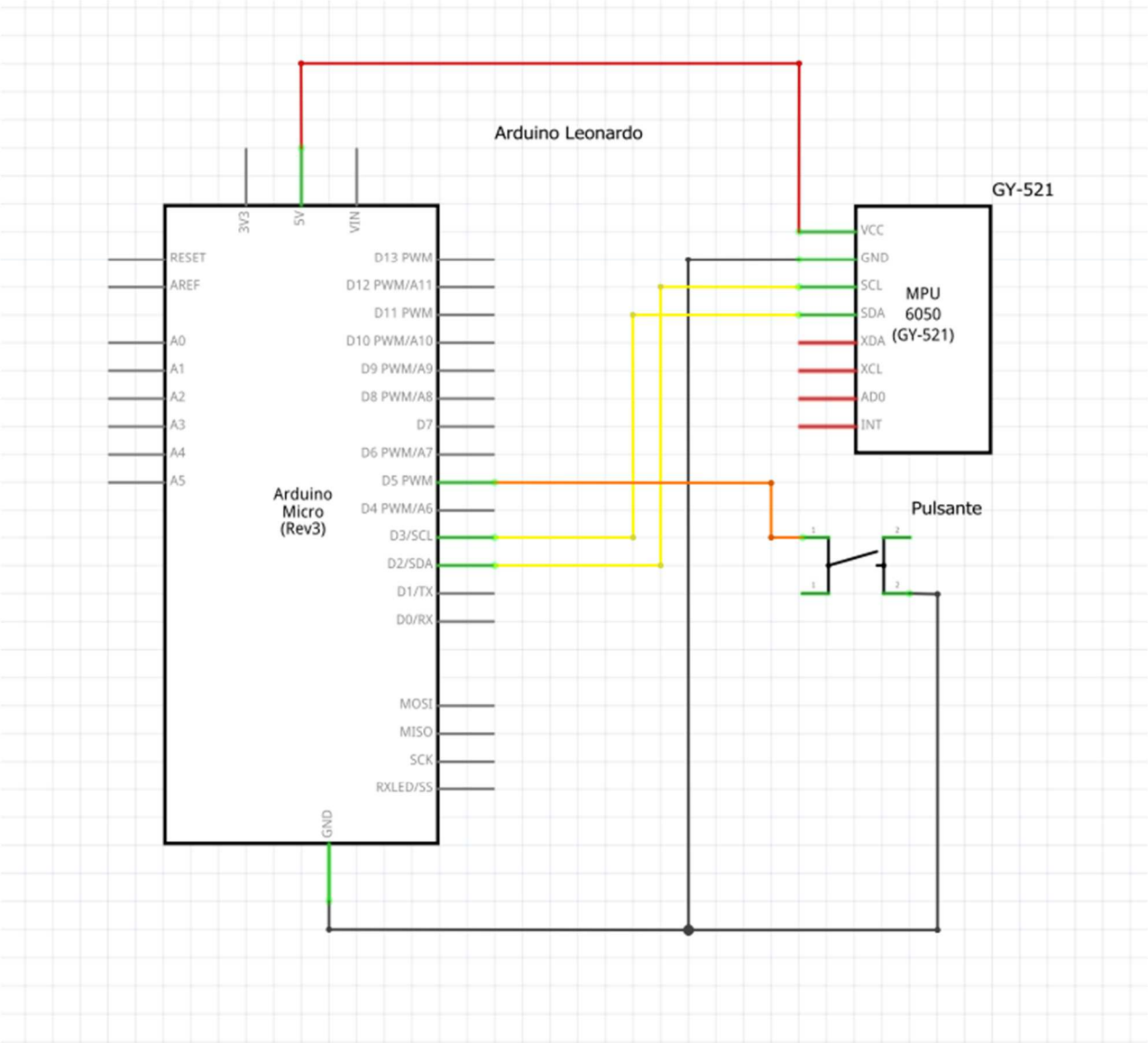
- MagicCap
- PanTilt

MagicCap

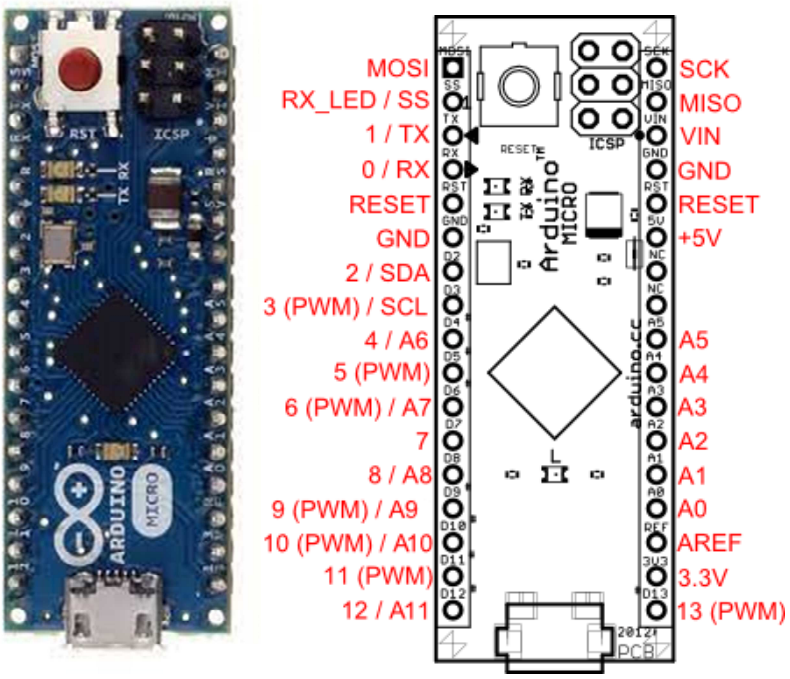
Componenti Utilizzati:

- Arduino Micro (Leonardo)
- GY-521 (accelerometro)
- Pulsanti
- Circuito stampato
- Struttura (cappellino + contenitore del circuito)

Schema Elettrico:



Microcontrollore Arduino Micro:



La Micro è una scheda microcontrollore basata su ATmega32u4. Dispone di 20 pin di ingresso / uscita digitali (di cui 7 possono essere utilizzati come uscite PWM e 12 come ingressi analogici), un oscillatore a cristallo da 16 MHz, una connessione micro USB, un'intestazione ICSP e un pulsante di ripristino. Contiene tutto il necessario per supportare il microcontrollore; basta collegarlo a un computer con un cavo micro USB per iniziare. Ha un fattore di forma che gli consente di essere facilmente posizionato su una breadboard. La scheda Micro è simile ad Arduino Leonardo in quanto l'ATmega32u4 ha una comunicazione USB incorporata, eliminando la necessità di un processore secondario. Ciò consente al Micro di apparire a un computer collegato come un mouse e una tastiera, oltre a una porta seriale / COM virtuale (CDC).

Quindi i motivi che ci hanno spinto ad usare questo controllore sono:

1) Perché è di piccole dimensioni così da non pesare su capo e collo.
2) Perché il computer lo riconosce come un mouse.

Librerie Arduino: mouse e tastiera

Queste librerie di base consentono a una scheda basata su 32u4 o una scheda Due e Zero di apparire come un mouse e / o una tastiera nativi su un computer collegato.

Per usare questa libreria:

#include <Keyboard.h>

#include <Mouse.h>

Mouse

Le funzioni del mouse consentono a Leonardo, Micro o Due di controllare il movimento del cursore su un computer collegato. Quando si aggiorna la posizione del cursore, è sempre relativa alla posizione precedente del cursore. Qui di seguito alcuni esempi di codice:

Mouse.begin ()

Mouse.click ()

Mouse.end ()

Mouse.move ()

Mouse.press ()

Mouse.release ()

Mouse.isPressed ()

Tastiera

Le funzioni della tastiera consentono a Leonardo, Micro o Due di inviare sequenze di tasti a un computer collegato. Non tutti i possibili caratteri ASCII, in particolare quelli non stampabili, possono essere inviati con la libreria Keyboard. La libreria supporta l'uso di tasti di modifica. I tasti modificatori cambiano il comportamento di un altro tasto quando vengono premuti contemporaneamente. In seguito alcuni esempi di codice:

Keyboard.begin ()

Keyboard.end ()

Keyboard.press ()

Keyboard.print ()

Keyboard.println ()

Keyboard.release ()

Keyboard.releaseAll ()

Keyboard.write ()

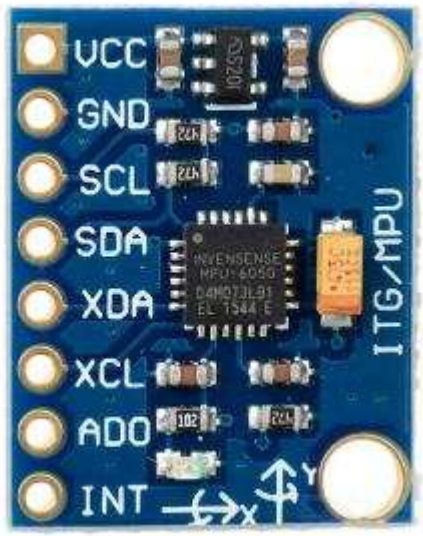
Acquisizione Dati

Ogni circuito elettronico in grado di comunicare con il mondo esterno, ha al suo interno un sistema di acquisizione e distribuzione dati. Questi sistemi hanno il compito di prelevare dal mondo esterno una grandezza fisica variabile, elaborarla ed infine digitalizzarla al fine di renderla adatta ad essere utilizzata in svariati modi dall'elaboratore, che è solitamente un PIC, un PLC o un microcontrollore più o meno elaborato.

Nello schema a blocchi riportato sopra, possiamo vedere l'acquisizione dati del sensore che abbiamo utilizzato, dove:

- La grandezza fisica è l'accelerazione.
- Il sensore è l'accelerometro
- Il circuito di condizionamento, il sample and hold e l'ADC sono integrati nel sistema.
- Il microcontrollore è l'Arduino micro.

Giroscopio GY-521



Caratteristiche del sensore:

- Alimentazione a 5Vcc
- Fondo scala giroscopio selezionabile tra ± 250, ± 500, ± 1000, e ± 2000°/sec
- Fondo scala accelerometro selezionabile tra ± 2g, ± 4g, 8g ± e ± 16g
- Interfaccia I2C
- Assorbimento durante la misura inferiore a 10mA

Abbiamo utilizzato questo sensore perché:

1) Leggero e compatto, così da non arrecare fastidio al capo e al collo.
2) Miglio rapporto qualità-prezzo
3) Condizionamento del sensore e convertitore analogico-digitale già integrato nel sistema.

Tali sensori sono utilizzati per robot capaci di stare in equilibrio, UAV, smartphone, nei dispositivi indossabili, nei controller e così via. I sensori IMU ci aiutano a stabilire la posizione di un oggetto e trovarne le coordinate (l'unità di misura utilizzata molto spesso sono i gradi) nelle tre dimensioni.

Il sensore GY-521 è sviluppato dalla società InvenSense e contiene al suo interno accelerometri e giroscopi MEMS (Micro Electro-Mechanical System). Inoltre contiene anche un convertitore analogico digitale a 16 bit per ogni canale.

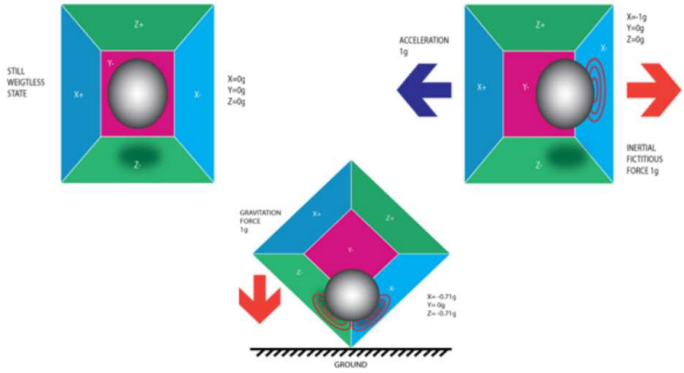
Per tal motivo rileva contemporaneamente l'inerzia sugli assi x, y e z. Per interfacciarsi con Arduino il sensore utilizza una connessione I2C-bus.

I sensori IMU sono composti da due o più componenti. Tra questi vi è l'accelerometro, il giroscopio, il magnetometro e l'altimetro.

Il sensore GY-521 è in grado di fornire ben 6 valori come output. Tre di questi saranno ricavati dall'accelerometro e 3 dal giroscopio. Accelerometro e giroscopio, sono entrambi contenuti all'interno di un singolo chip.

Come funziona l'accelerometro

Per il cappellino abbiamo sfruttato l'accelerometro.



Un accelerometro funziona sfruttando il principio dell'effetto piezoelettrico. Basta Immaginare una scatola di forma cubica con una piccola sferetta all'interno come nell'immagine a lato. Le pareti della scatola sono fatte di materiale cristallino piezoelettrico. Quando si ruota il box la sferetta è forzata a muoversi, ad opera della forza di gravità, nella direzione in cui viene inclinata la scatola. Le pareti contro cui la sferetta collide, creano delle leggerissime scariche di piezoelettricità.

La scatola è composta da un totale di 3 paia di pareti opposte. Ogni paio corrisponde ad un asse nello spazio tridimensionale: secondo le assi X, Y, Z. In base alla corrente piezoelettrica prodotta dalle pareti cristalline si può determinare la direzione dell'inclinazione.

Un'improvvisa accelerazione provoca uno shock e una compressione della struttura del cristallo piezoelettrico che reagisce generando un segnale elettrico ben preciso. Rilevando questo segnale elettrico ed elaborandolo adeguatamente, l'accelerometro è dunque in grado di determinare da quale direzione giunga la spinta acceleratrice e quale sia la sua forza.

I movimenti saranno registrati esclusivamente su un piano orizzontale (o verticale) e dunque bidimensionale.

Per la digitalizzazione dei valori viene utilizzato un ADC a 16 bit, Gli intervalli di uscita a fondo scala possibili sono: +/- 2g, +/- 4g, +/- 8g, +/- 16g.

In posizione normale, cioè quando il dispositivo è posizionato su una superficie piana, i valori sono 0 g sull'asse x, 0 g sull'asse y e +1 sull'asse z.

Comunicazione I^2C (per GY-521)

I^2C (abbreviazione di Inter Integrated Circuit), è un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati.

Il classico bus I^2C è composto da almeno un master ed uno slave.

La situazione più frequente vede un singolo master e più slave; possono tuttavia essere usate architetture multimaster e multislave in sistemi più complessi.

Un suo limite è la velocità di comunicazione anche se lo standard iniziale è stato comunque aggiornato sotto questo aspetto.

L'I2C ha 7 bit di indirizzo (B1 è il bit più significativo, B7 il meno significativo) e quindi 128 possibili indirizzi diversi (detti nodi). Di questi però 16 sono riservati, quindi i dispositivi che possono essere collegati sullo stesso bus sono al massimo 112. Le velocità di trasmissione nel modo standard sono di 100 kbit/s e 10 kbit/s (velocità del low-speed mode) ma nulla impedisce di scendere a velocità più basse.

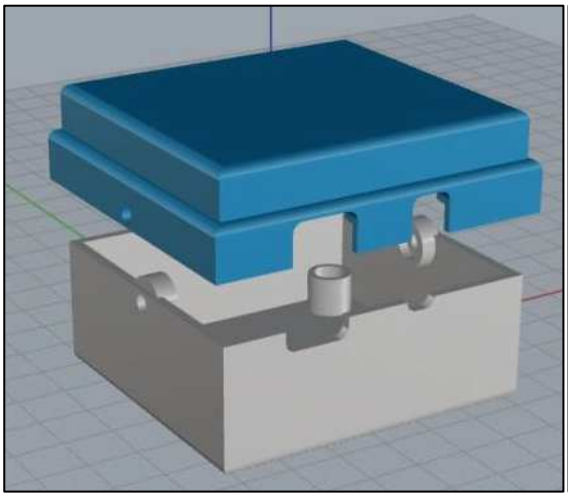
Revisioni del I2C hanno introdotto dispositivi con velocità di 400 kbit/s (detto fast mode) e 3,4 Mbit/s (detto High Speed mode). Anche le possibilità di indirizzamento di dispositivi sono state ampliate a 10 bit.

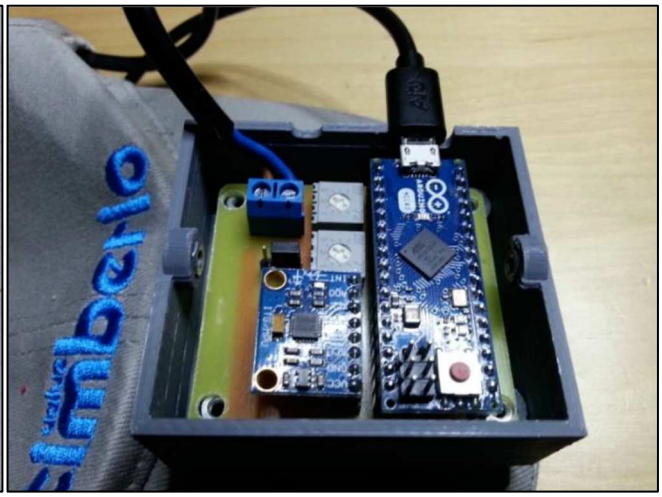
Il massimo numero di nodi oltre ad essere limitato dal numero di indirizzi possibili diversi è limitato dalle capacità parassite introdotte da ciascun dispositivo. La capacità totale presentata da SDA e da SCL deve essere limitata a 400 pF (i resistori di pull-up dovranno essere dimensionati su questo dato, cioè la massima capacità dovuta alla somma dei dispositivi connessi, oltre al limite della corrente che ogni dispositivo può assorbire al livello logico basso).

Pulsante

Il pulsante tra i denti può essere utilizzato in maniera molto semplice: con una leggera pressione vado a emulare il click del mouse sinistro, tramite una pressione più prolungata quello destro. Tutto questo è stato realizzato tramite software.

Contenitore e PCB





Il contenitore è stato realizzato con una stampante 3D, utilizzando per la progettazione un CAD 3D, con d esempio Tinkercad. Il sistema è stato montato su una PCB stampata e fissato sull'aletta del cappellino, si può osservare il cavo di connessione USB, verso il PC, e quello tramite cui si collega il pulsante.

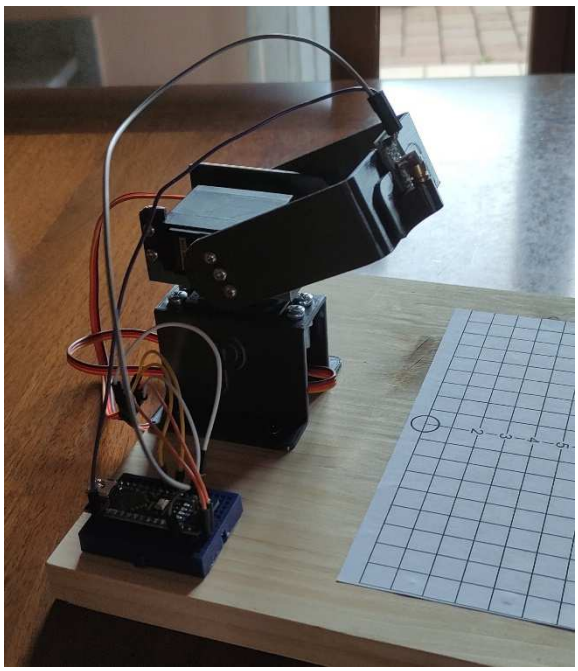
Sketch (Arduino IDE)

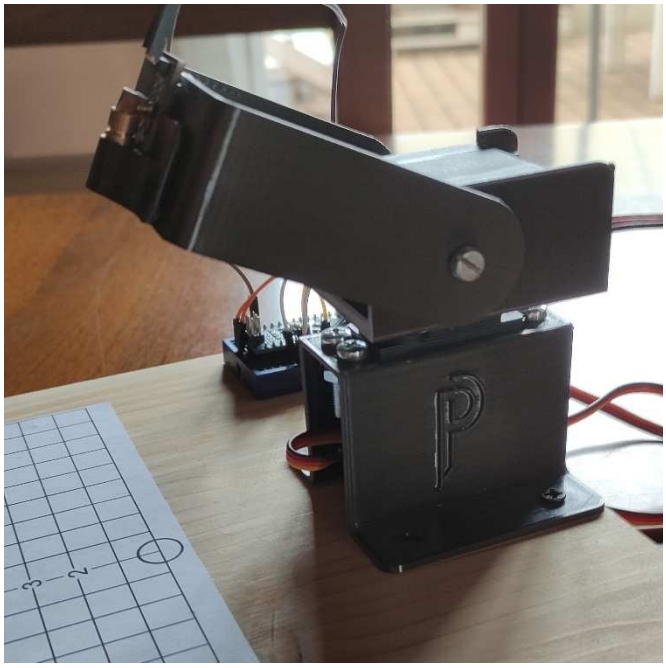
```
#include <Mouse.h>
#include <SPI.h>
#include <Wire.h>

#define MPU 0x68  // I2C address of the MPU-6050

unsigned long durata;
int stato5;
double AcX, AcY, AcZ;
int Pitch, Roll;

int clickPinDig = 5;   //pin di ingresso del puls

const int enableMousePin = 6;   //pin per l'enable del mouse

unsigned long Millis;  //Variabile per il Millis

void setup()
{
Serial.begin(9600);
init_MPU();  //Inizializzazione MPU6050

// pinMode(enableMousePin,INPUT);
pinMode(clickPinDig,INPUT);
digitalWrite(clickPinDig,HIGH);

// initialize mouse control:
if(digitalRead(enableMousePin)==LOW)
```

```
{
Mouse.begin();
}
}

void loop()
{
FunctionsMPU();  //Acquisisco assi AcX, AcY, AcZ.

Roll = FunctionsPitchRoll(AcX, AcY, AcZ) * 4;  //Calcolo angolo Roll
Pitch =  -FunctionsPitchRoll(AcY, AcX, AcZ) * 4;  //Calcolo angolo Pitch
//moltiplicando per la costante (4) cambia la sensibilità

delay(100);

if(digitalRead(enableMousePin) == LOW)
{
Mouse.move(0.5 * Pitch, 0.5 * Roll, 0);
}

while(digitalRead(2) == LOW)
{
delay(100);
Millis = Millis + 100;

Serial.print("ms = ");
Serial.println(Millis);
}

//Emulazione del mouse con il pulsante con comando Millis
if(Millis >= 500)
{
Mouse.click(MOUSE_LEFT);
Mouse.click(MOUSE_LEFT);
delay(150);
}

else if(Millis >= 100)
{
Mouse.click(MOUSE_RIGHT);
delay(150);
}

Millis = 0;
}

void init_MPU()  //Inizializza il GY-521
{
Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B);  // PWR_MGMT_1 register
Wire.write(0);     // set to zero (wakes up the MPU-6050)
Wire.endTransmission(true);
delay(1000);
}

//Funzione per il calcolo degli angoli Pitch e Roll
double FunctionsPitchRoll(double A, double B, double C){
double DatoA, DatoB, Value;
DatoA = A;
DatoB = (B*B) + (C*C);
DatoB = sqrt(DatoB);

Value = atan2(DatoA, DatoB);
Value = Value * 180/3.14;
```

```
return (int)Value;
}

//Funzione per l'acquisizione degli assi X,Y,Z del MPU6050
void FunctionsMPU()
{
Wire.beginTransmission(MPU);
Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU,6,true);  // request a total of 14 registers
AcX = Wire.read()<<8 | Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8 | Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8 | Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
}
```

PanTilt



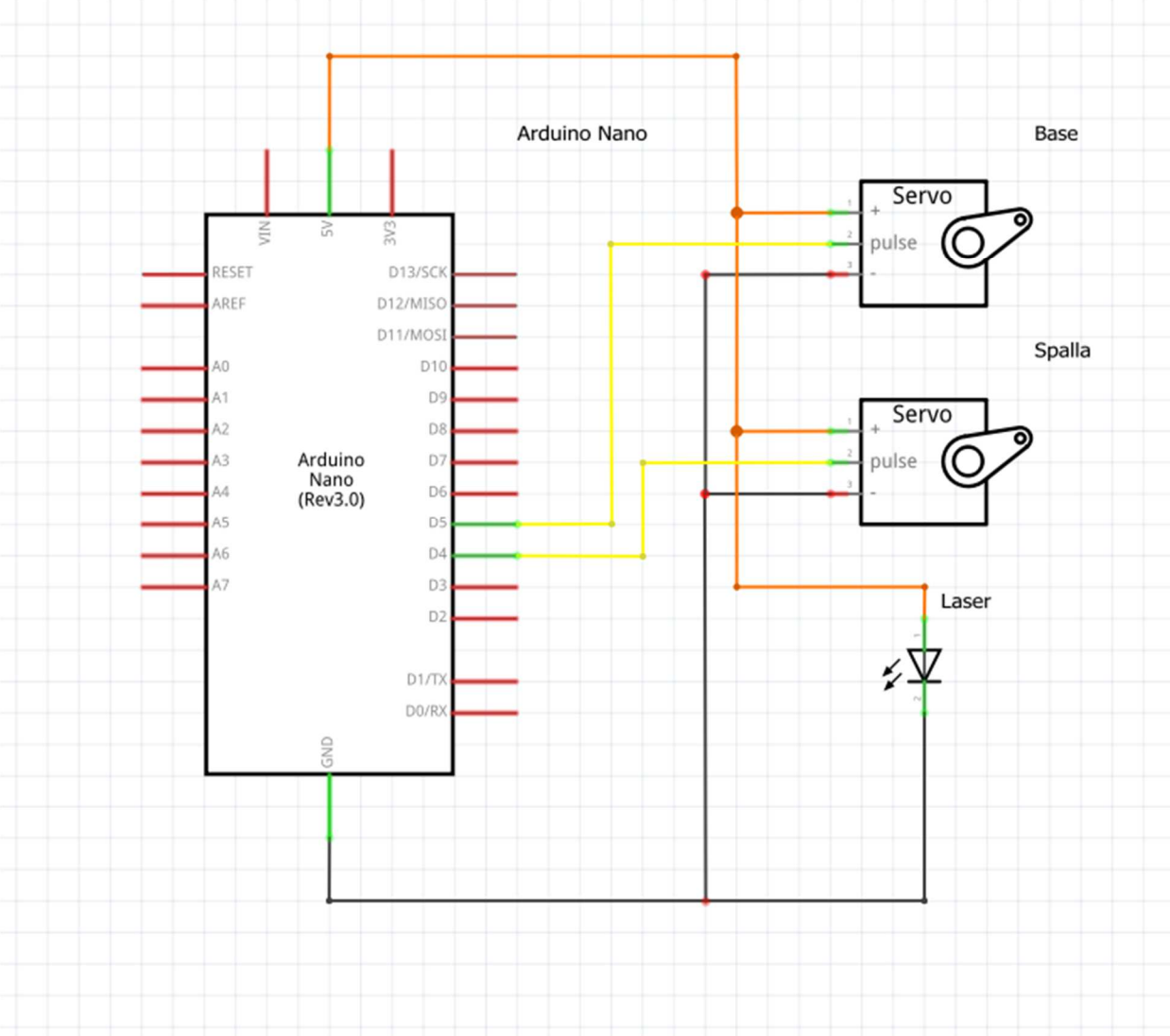


Il PanTilt è una struttura che abbiamo creato durante l'anno scolastico in DAD non potendo usufruire del braccio robotico che si trova in laboratorio. Funziona inviando delle coordinate tramite l'applicazione sviluppata in C#. L'obbiettivo del suo utilizzo è lo spostamento della struttura tramite 2 servomotore ed il puntamento sulla scacchiera nella coordinata scelta con un modulo laser.

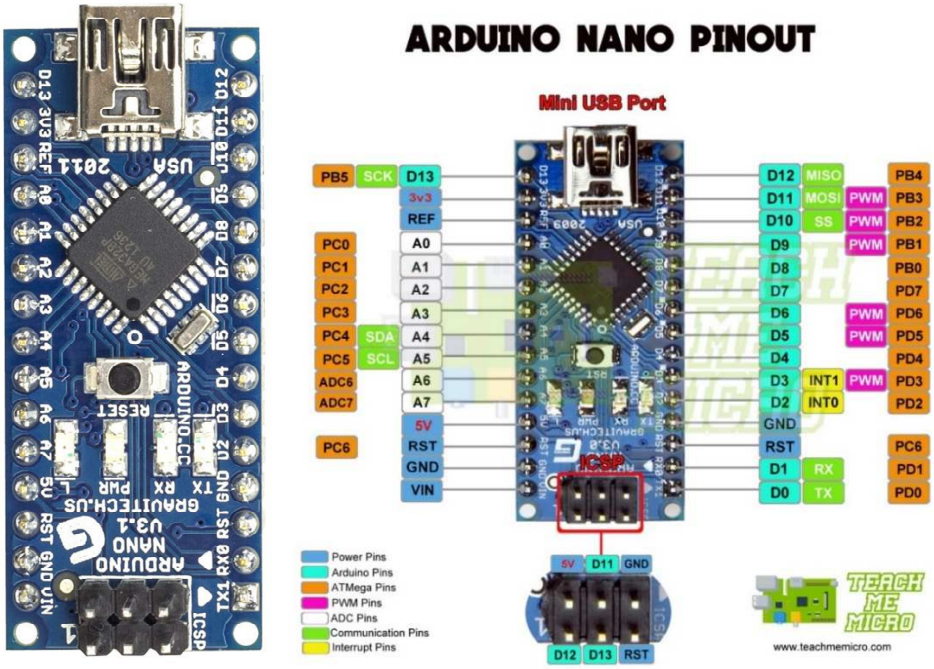
Componenti Utilizzati:

- Microcontrollore: Arduino Nano
- 2 Servo Motore
- 1 Modulo Laser
- Struttura Pan and Tilt (Stampata con la stampante 3D)
- Software: Visual Studio, Arduino

Schema Elettrico

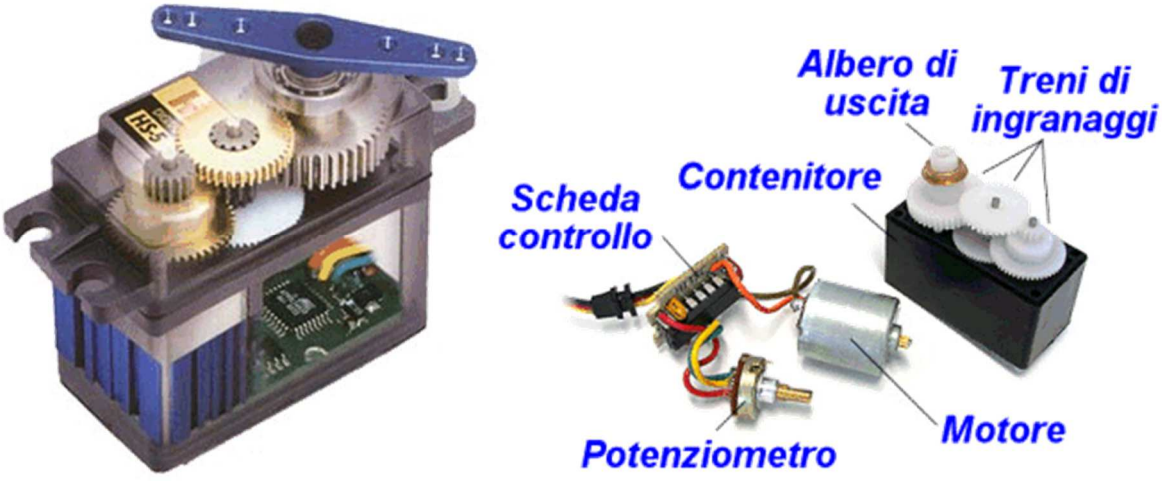


Arduino nano



Arduino Nano è una scheda piccola, completa e adatta alla breadboard, basata su ATmega328 (Arduino Nano 3.x). Ha più o meno le stesse funzionalità di Arduino UNO, ma in un pacchetto diverso. Manca solo un jack di alimentazione CC e funziona con un cavo USB Mini-B invece di uno standard.
Lo abbiamo utilizzato per ricevere i dati da C# cosi da controllare i servomotori.

Servo motore

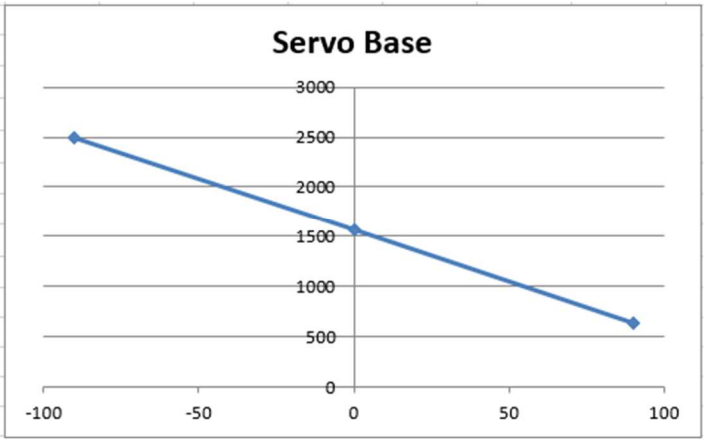


Nella robotica per gli azionamenti, sono molto utilizzati i servomotori. Di solito questi si presentano come piccoli contenitori di materiale plastico da cui fuoriesce un perno in grado di ruotare in un angolo compreso tra 0 e 180° mantenendo stabilmente la posizione raggiunta. Per ottenere la rotazione del perno è utilizzato un motore a corrente continua e un meccanismo di demoltiplica che consente di aumentare la coppia in fase di rotazione. La rotazione del motore è effettuata tramite un circuito di controllo interno in grado di rilevare l'angolo di rotazione raggiunto dal perno.
All'interno del servo è presente un motore, una serie di ingranaggi che riducono la velocità del motore, un circuito di controllo e un potenziometro.
Il motore e il potenziometro sono collegati al circuito di controllo e l'insieme di questi tre elementi definisce un sistema di feedback ad anello chiuso. Il circuito e il motore vengono alimentati da una tensione continua stabilizzata, in genere di valore compreso tra 4,8 V e 6,0 V, anche se molti motori sono in grado di accettare input di alimentazione fino a 7,2 V.
Per far girare il motore bisogna inviare un segnale digitale al circuito di controllo, in questo caso dopo i calcoli è inviato dall'Arduino nano. Così si attiva il motore che, attraverso una serie di ingranaggi, è collegato al potenziometro. La posizione dell'albero del potenziometro indica una misura della posizione dell'albero motore del servo. Quando il potenziometro raggiunge la posizione desiderata, il circuito di controllo spegne il motore.
È facile dedurre che i servomotori vengono progettati in genere per effettuare una rotazione parziale piuttosto che impostare un moto rotatorio continuo, come nel caso di un motore in continua o passo-passo. Anche se è possibile configurare un servo R/C perché ruoti in modo continuo, l'impiego fondamentale di un servo consiste nel raggiungere una posizione accurata dell'albero del motore, con movimenti compresi nell'intervallo tra 90° e 180'. Anche se questo movimento non sembra considerevole, può risultare più che sufficiente per manovrare un robot, per sollevare e abbassare le gambe, per ruotare un sensore che deve esaminare ciò che le circonda e molto altro ancora. La rotazione precisa di un angolo da parte di un servo in risposta a determinati segnali digitali rappresenta una delle funzionalità più sfruttate in tutti i campi della robotica.
In questo caso li abbiamo utilizzati per lo spostamento della struttura PanTilt, così da far puntare il laser nella posizione desiderata, per farlo abbiamo dovuto "tarare" i servo.
Per taratura si intende la determinazione della corrispondenza tra angolo del giunto e durata degli impulsi che l'interfaccia deve fornire al servomotore. Per poterla

effettuare occorre predisporre un programma di taratura, che si può realizzare con Arduino, oppure con C#.
Questi dati si ottengono per tentativi, modificando tramite il programma il valore degli impulsi, fino a trovare le posizioni che ci interessano.
Con l'impiego di un programma di calcolo, in questo caso Excel, è possibile ricavare da ogni tabella un grafico che permetta di costruire una equazione che leghi le variabili in gioco:

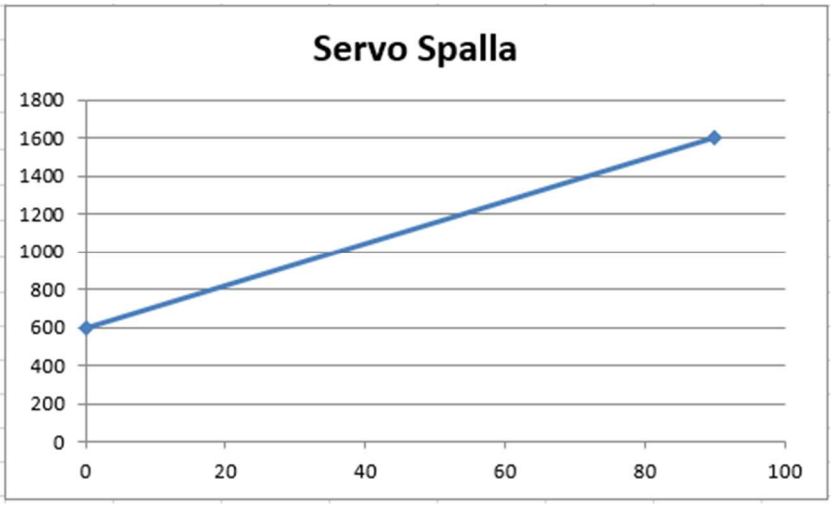
Servo Base	Gradi	Impulsi
Dx	90	630
Centro	0	1570
Sx	-90	2500

m	q
-10,3889	1570



Servo Spalla	Gradi	Impulsi
Alto	90	1600
Basso	0	600

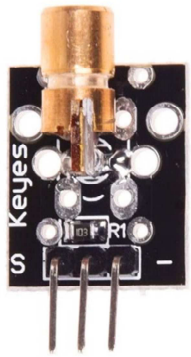
m	q
11,11111	600



In base ai dati ipotizzati si ottengono i dati del tipo in figura, da cui si nota una relazione lineare tra angolo ed impulsi. Questa è una relazione che si può ipotizzare vera per la maggior parte dei servomotori.

Le relazioni sono:
Impulsi = angolo * m + q
q = Intercetta della retta
m = (P2y - P1y) / (P2x - P1x)

Modulo Laser

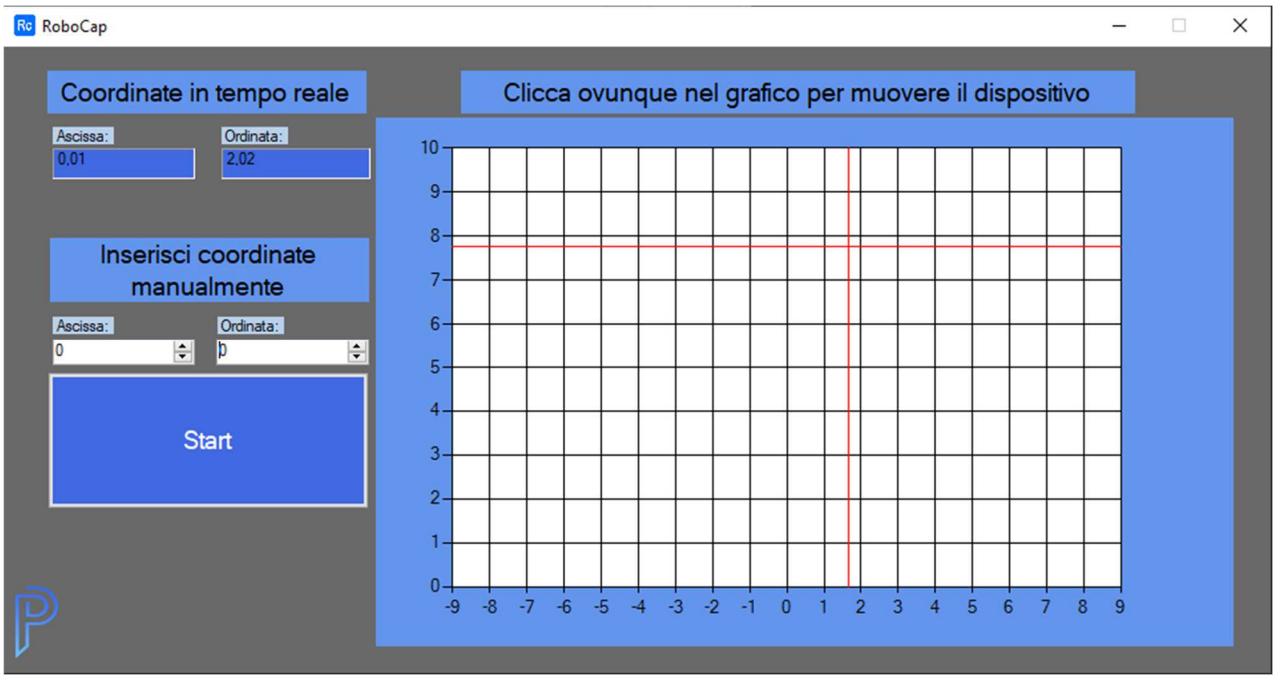


Il modulo KY-008 è composto da un piccolo diodo laser e una resistenza per limitare la corrente assorbita dal diodo. Il suo utilizzo è molto semplice in quanto basta solamente alimentarlo collegandolo all'Arduino. Lo abbiamo utilizzato per far puntare al PanTilt sulla scacchiera, la coordinata desiderata.

Struttura pan tilt con stampante 3D

Le parti principali della struttura sono realizzate con un cad 3D come Tinkercad, e stampate in PLA con la stampante 3D.

Sketch Visual Studio



Lo sketch di visual studio è composto da due parti principali. Nella parte grafica, tramite la scacchiera virtuale, si scelgono le coordinate x e y, la parte di codice calcola i valori per lo spostamento dei servo, per poi inviarli tramite la seriale ad Arduino.

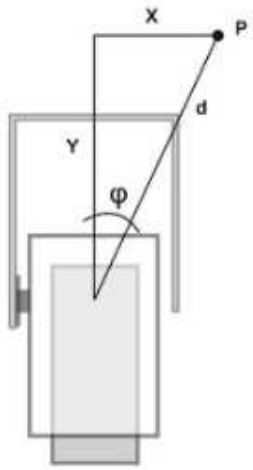
A lato la spiegazione dello sketch tramite un flow chart.

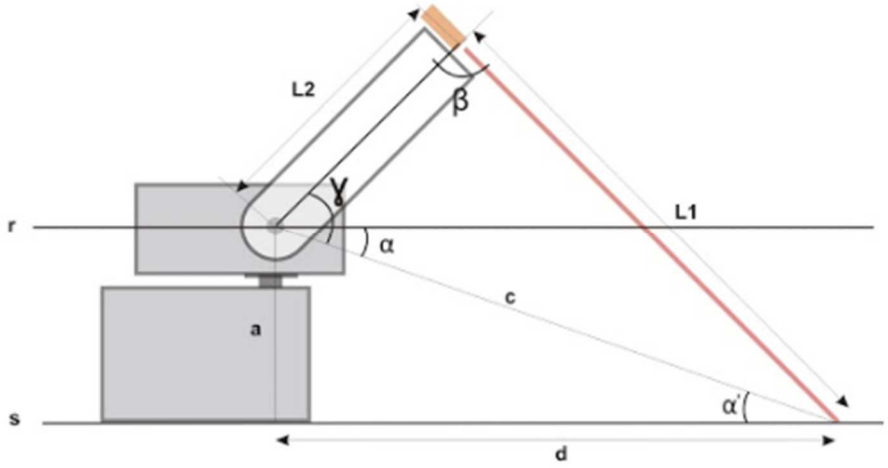

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Sketch_Pan_Tilt_con_Canvas_Virtuale
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
        double l2 = 7;    //Misura dall'ingrsnaggio del servo spalla al laser
        double a = 8.25;  //Altezza da terra alll'ingranaggio del servo spalla, Tarato 7 + 1.25
        double dB = 6.5;  //Distanza della scacchiera all'ingranaggio del servo di base

        //base
        double mBase = -10.3889;  //coefficente angolare del servo di base (Taratura)
        double qBase = 1560;      //Intercetta del servo di base (Taratura)

        //servoSpalla
        double mSpalla = 11.1111;  //coefficente angolare del servo spalla (Taratura)
        double qSpalla = 600;      //Intercetta del servo spalla (Taratura)

        private void cmdButton_Click_1(object sender, EventArgs e)
        {
            double x = double.Parse(nud1.Text);   //Coordinata x dal numeric up down
            double y = double.Parse(nud2.Text);   //Coordinata y dal numeric up down

            double d = Math.Sqrt(Math.Pow(x, 2) + Math.Pow((y + dB), 2));  //Distanza dal giunto
di base al valore massimo di y
            double c = Math.Sqrt(Math.Pow(d, 2) + Math.Pow(a, 2));         //Ipotenusa c
            double l1 = Math.Sqrt(Math.Pow(c, 2) - Math.Pow(l2, 2));          //Link 2, giunto
variabile relativo al raggio laser

            double angoloBase = Math.Atan(x / (y + dB));  //Trovo l'angolo di base
            double angoloSpalla = Math.Atan(l1 / l2);     //Trovo l'angolo spalla
            double alfaRadianti = Math.Atan(a / d);

            angoloBase = angoloBase * 57.296;        //Conversione da radianti in gradi
            angoloSpalla = angoloSpalla * 57.296;    //Conversione da radianti in gradi
            alfaRadianti = alfaRadianti * 57.296;    //Conversione da radianti in gradi

            angoloSpalla = angoloSpalla - alfaRadianti;

            int impulsiBase = (int)(angoloBase * mBase + qBase);           //Calcolo dell'impulso
per il servo di base grazie alla taratura
            int impulsiSpalla = (int)(angoloSpalla * mSpalla + qSpalla);   //Calcolo dell'impulso
per il servo spalla grazie alla taratura

            string Base = impulsiBase.ToString();       //Conversione degli interi in stringhe
            string Spalla = impulsiSpalla.ToString();   //Conversione degli interi in stringhe

            Base = Base.PadLeft(4, '0');        //Compatto il dato
            Spalla = Spalla.PadLeft(4, '0');    //Compatto il dato

            lblBase.Text = impulsiBase.ToString();
            lblSpalla.Text = impulsiSpalla.ToString();

            serialPort.Open();                       //Apertura della seriale
            serialPort.Write(Base + "B");             //Manda il dato ad arduino per il servo di
base
            serialPort.Write(impulsiSpalla + "S");   //Manda il dato ad arduino per il servo spalla
            serialPort.Close();                      //Chiude la seriale
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Proprietá del grafico
            chart1.ChartAreas[0].AxisX.Minimum = -9;  //Proprietá asse X (Min)
            chart1.ChartAreas[0].AxisX.Maximum = 9;   //Proprietá asse X (Max)
            chart1.ChartAreas[0].AxisX.Interval = 1;  //Intervallo asse X

            chart1.ChartAreas[0].AxisY.Minimum = 0;   //Proprietá asse Y (Min)
            chart1.ChartAreas[0].AxisY.Maximum = 10;  //Proprietá asse Y (Max)
            chart1.ChartAreas[0].AxisY.Interval = 1;  //Intervallo asse Y

            chart1.Series["Series1"].Points.AddXY(9, 0);
```

```
            chart1.Series["Series1"].BorderWidth = 2;
        }

        private void chart1_MouseMove_1(object sender, MouseEventArgs e)
        {
            //Ottengo le coordinate dal grafico virtuale
            Point mousePoint = new Point(e.X, e.Y);

            chart1.ChartAreas[0].CursorX.Interval = 0;
            chart1.ChartAreas[0].CursorY.Interval = 0;

            chart1.ChartAreas[0].CursorX.SetCursorPixelPosition(mousePoint, true);
            chart1.ChartAreas[0].CursorY.SetCursorPixelPosition(mousePoint, true);
        }

        private void chart1_MouseClick(object sender, MouseEventArgs e)
        {
            double xCo = chart1.ChartAreas[0].AxisX.PixelPositionToValue(e.X);
            double yCo = chart1.ChartAreas[0].AxisY.PixelPositionToValue(e.Y);

            double x = Math.Round(xCo, 2);
            double y = Math.Round(yCo, 2);

            lblBase.Text = x.ToString();
            lblSpalla.Text = y.ToString();

            double d = Math.Sqrt(Math.Pow(x, 2) + Math.Pow((y + dB), 2));
            double c = Math.Sqrt(Math.Pow(d, 2) + Math.Pow(a, 2));
            double l1 = Math.Sqrt(Math.Pow(c, 2) - Math.Pow(l2, 2));

            double angoloBase = Math.Atan(x / (y + dB));
            double angoloSpalla = Math.Atan(l1 / l2);
            double alfaRadianti = Math.Atan(a / d);

            angoloBase = angoloBase * 57.296;
            angoloSpalla = angoloSpalla * 57.296;
            alfaRadianti = alfaRadianti * 57.296;

            angoloSpalla = angoloSpalla - alfaRadianti;

            int impulsiBase = (int)(angoloBase * mBase + qBase);
            int impulsiSpalla = (int)(angoloSpalla * mSpalla + qSpalla);

            string Base = impulsiBase.ToString();
            string Spalla = impulsiSpalla.ToString();

            Base = Base.PadLeft(4, '0');
            Spalla = Spalla.PadLeft(4, '0');

            lblBase.Text = impulsiBase.ToString();
            lblSpalla.Text = impulsiSpalla.ToString();

            serialPort.Open();
            serialPort.Write(Base + "B");
            serialPort.Write(impulsiSpalla + "S");
            serialPort.Close();
        }
    }
}
```

Calcoli





Di seguito sono riportati i calcoli per trovare gli angoli grazie ai quali si possono calcolare gli impulsi per i servomotori:

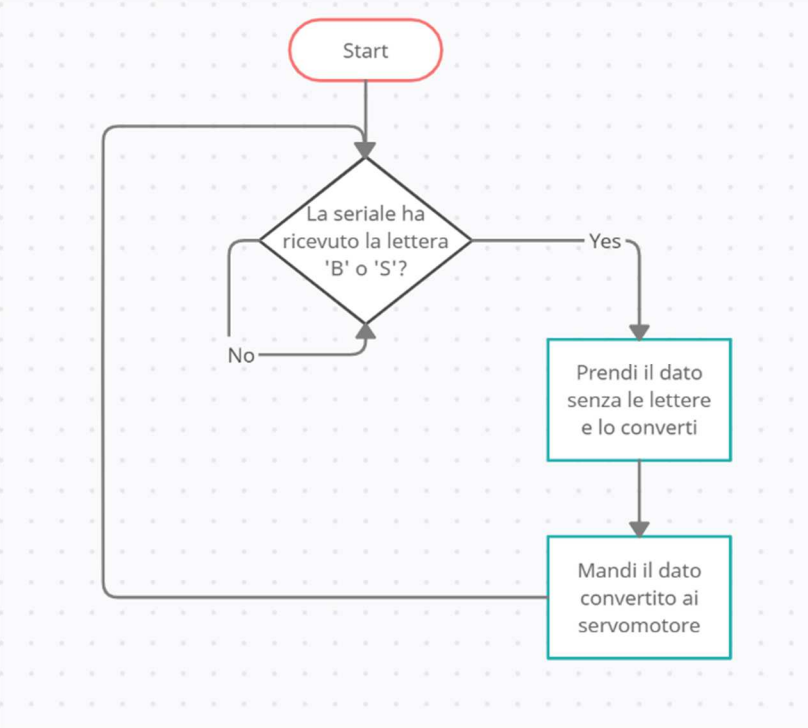
- $d = \sqrt{x^2 + (y + dB)^2}$ dove x e y sono le coordinate e dB è la distanza del giunto di base dalla scacchiera.
- $c = \sqrt{d^2 + a^2}$
- $l1 = \sqrt{c^2 + l2^2}$
- $\varphi = arctg(\frac{x}{(y + dB)})$
- $\gamma = arctg(\frac{l1}{l2})$
- $\alpha = arctg(\frac{a}{d})$

Dopo aver trovato gli angoli in radianti bisogna convertirli in gradi, per poi usare la formula: angolo * m + q per trovare gli impulsi corretti per i servomotore.

Sketch Arduino IDE

Arduino nano è usato per mandare i valori per far spostare i Servomotore nella posizione scelta, dopo averli ricevuti dall'applicazione di visual studio.

A lato la spiegazione dello sketch tramite un flow chart.




```
#include <Servo.h>  //Libreria per i servo arduino

Servo servoBase;    //Inizializzi servo base
Servo servoSpalla;  //Inizializzi servo spalla

String inputBase;
String inputSpalla;
int impulsiBase;
int impulsiSpalla;

void setup()
{
Serial.begin(9600);     //Inizializza seriale
servoBase.attach(5);    //Servo base attach al pin 5
servoSpalla.attach(3);  //Servo spalla attach al pin 3
}

void loop()
{
if(Serial.available() > 0)  //Se la seriale riceve qualcosa
{
inputBase = Serial.readStringUntil('B');    //Legge fino a che non riceve il carattere 'B'
inputSpalla = Serial.readStringUntil('S');  //Legge fino a che non riceve il carattere 'S'

inputBase = inputBase.substring(0,4);     //Decompatta la stringa
inputSpalla = inputSpalla.substring(0,4);  //Decompatta la stringa

impulsiBase = inputBase.toInt();     //Stringa to intero
impulsiSpalla = inputSpalla.toInt();  //Stringa to intero
}

servoBase.writeMicroseconds(impulsiBase);     //Scrive sui servo
servoSpalla.writeMicroseconds(impulsiSpalla);  //Scrive sui servo

Serial.print(impulsiBase);
Serial.print("   ");
Serial.println(impulsiSpalla);
delay(200);
}
```

Conclusioni:

Il nostro obbiettivo nel creare questo progetto è di dare un esempio pratico di come anche una persona disabile possa integrarsi nel mondo del lavoro, in aziende robotizzate e all'avanguardia nei settori della robotica e dell'automazione. Il RoboCap è solamente una simulazione, ma nell'uso comune il cappellino può essere collegato ad un braccio robotico aziendale che permetta ad una persona diversamente abile di integrarsi anche nel mondo del lavoro. Questo è solamente un punto di inizio, il nostro progetto può essere sicuramente migliorato.

Sitografia:

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

https://www.arduino.cc/