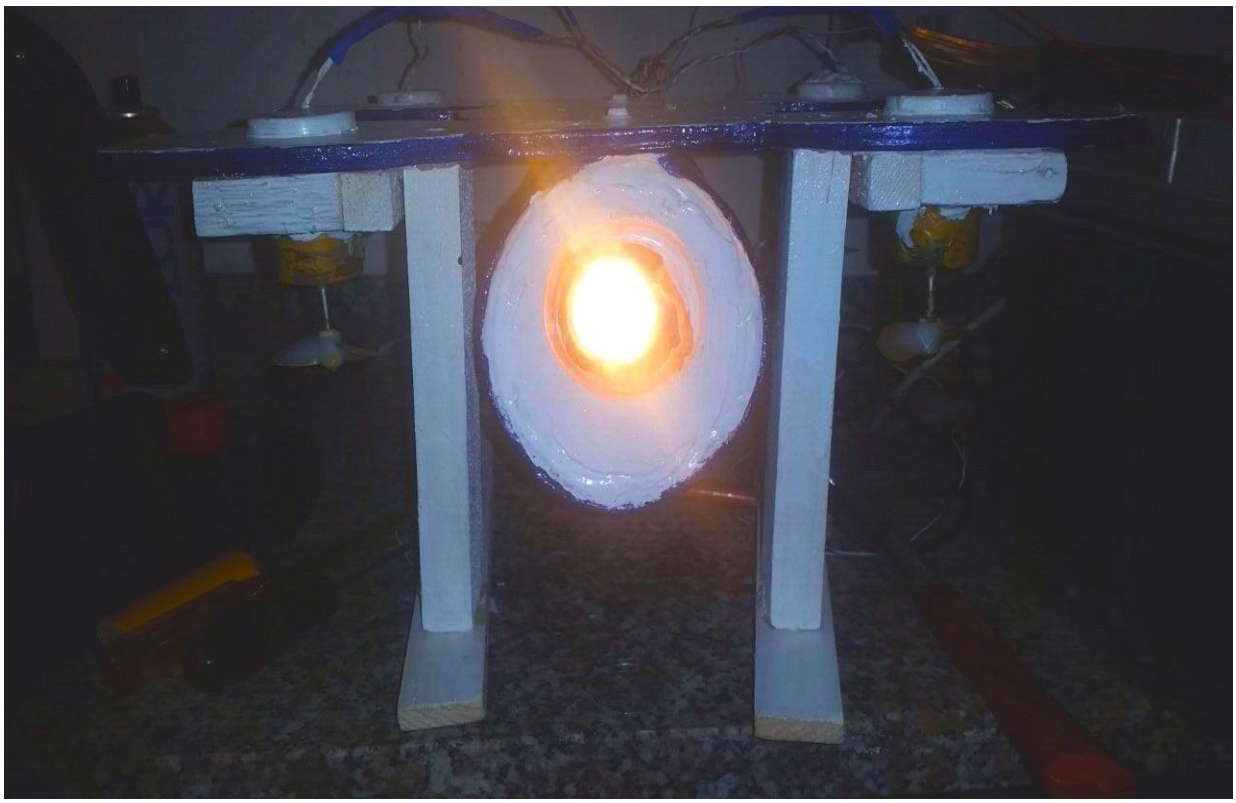
Intestazione

Franco	Bassanetti	4Q	costruttore e realizzatore
Francesco	Bella	4Q	costruttore e realizzatore
Maria	Giraldo	4Q	sviluppatrice
Lorenzo	Ravecca	4Q	analista
Simone	Piana	4Q	programmatore

Data di realizzazione : A.S 2014/2015

Titolo

Drone acquatico (batiscafo) sviluppato con Arduino e comandato con controllo remoto.



Abstract

Progetto sviluppato con la piattaforma Arduino, al fine di riprodurre un robot per eco scandagliare il fondale marino. Un esperimento con una durata di circa un anno ed ancora in via di sviluppo.

Introduzione/scopo dell'esperienza

Questo progetto consiste nella riproduzione di un drone acquatico o batiscafo, per lavori di recupero o assistenza di operazioni a basse profondità. Gli obiettivi principali nella realizzazione di questo progetto sono stati:

- -costruzione di una struttura leggera con materiali economici reperibili facilmente in commercio
- -implementazione degli integrati necessari per il completamento del progetto
- -programmazione in linguaggio microC della piattaforma Arduino

Trattazione teorica

Questo progetto si articola in due momenti:

1. La realizzazione della struttura (parte meccanica/fisica del batiscafo)
2. L'implementazione degli integrati e dei circuiti ci siamo basati su diversi datasheet.

La realizzazione della struttura si basa essenzialmente sul principio di Archimede: ogni corpo immerso parzialmente o completamente in un fluido (liquido o gas) riceve una spinta verticale dal basso verso l'alto, uguale per intensità al peso del fluido che occupa nel volume spostato.

La formula è la seguente: $FARC = Ps_{liq} \cdot Vimm$.

Per capire meglio il comportamento di un corpo immerso di un fluido abbiamo svolto vari test per comprendere come questo reagisse. Infine è stata aggiunta una luce nella posizione frontale (vedi foto copertina).

Strumenti utilizzati

Abbiamo lavorato utilizzando come strumenti principali due micro controllori Arduino UNO e un computer con Windows 8.1 come Sistema Operativo. Utilizzando microC come ambiente di programmazione, abbiamo creato un programma capace di governare il batiscafo e mantenerlo stabile durante il tragitto in acqua.
La componentistica utilizzata è:

- LB293 (ponte h)
- XBee (WI-FI)
- MPU-6050 (giroscopio)

in più, ovviamente, abbiamo utilizzato cavi di natura varia, di stagno per saldature ed legno per la struttura del batiscafo.

Procedimento

Prima fase: *creazione dell'esoscheletro.*
E' stato utilizzato un tubo in PVC del diametro di 10 cm e in seguito sono stati adattati alcuni pezzi di compensato marino che andranno a formare la parte superiore e i piedi d'appoggio.

Seconda fase: *movimenti e stabilità.*
In questa fase si è pensato all'implementazione dei motori che andranno a consentire lo spostamento in tutte le direzioni e la stabilita anche in caso di correnti marine.
Quattro motori sono stati montati nella parte superiore per consentire l'immersione e il ritorno in superficie nonché la stabilizzazione mediante un giroscopio (MPU 6050).
Invece i motori restanti sono stati collocati nella parte posteriore e, mediante software, consentono propulsione e virata.

La parte software che li gestisce è così strutturata:

1) Controllo mediante giroscopio della posizione e dell'inclinazione del mezzo.
2) Attivazione dei motori in caso di una inclinazione maggiore di 1.5/2 gradi.
3) Ristabilizzazione dopo la ricezione di un segnale inviato dall'utente.

Terza fase: *galleggiante di supporto marino.*
Per alloggiare il pacco batterie è stata costruita una boa di medie dimensioni utilizzando una struttura tubolare per il contenimento di materiali gassosi inerti in pressione, sulla quale è stata fissata una struttura esagonale in compensato da 4mm di spessore, che fungerà da contenitore per le batterie,il Arduino,le basette e/o le millefori.
Al centro della struttura è stato praticato un foro per facilitare il passaggio del cavo di comunicazione digitale (Arduino UNO) e del cavo coassiale per la trasmissione video del dispositivo di acquisizione d'immagini, resistente all'acqua.

Quarta fase: *prova pratica in piscina*
Il giorno 12 dicembre 2014 abbiamo collaudato il nostro mezzo nella piscina della scuola per verificarne le reali capacità di movimento. Per l'evento avevamo costruito un rudimentale joystick con una breadboard e dei pulsanti. Il testing si è concluso nel migliore dei modi.

Quinta fase: *upgrade dei componenti*
Dopo il testing e fino ad oggi, stiamo lavorando sul progetto per migliorare la stabilità e gli eventuali bug tecnici nel software. Attualmente abbiamo risolto tutti i problemi riguardanti l'impermeabilizzazione e stiamo migliorando la parte software.

Risultati

Durante le varie fasi di sviluppo e testing abbiamo riscontrato alcune difficoltà :
1)La parte più complicata è stata quella di reperire il materiale in maniera economica, proprio per questo è stato difficile il recupero delle strutture.
2)Il secondo inconveniente è stata la progettazione e realizzazione di un assetto idrodinamico e di un corretto bilanciamento anche in presenza di forti correnti marine.

Codice

```
// include librerie necessarie per l'uso del gy-521

#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL



// MPU6050 variabili di controllo
bool dmpReady = false;  // se diventa true l'inizializzazione con mpu6050 è
riuscita
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// variabili di orientamento/accellerazione etc...
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor
measurements
VectorInt16 aaWorld;    // [x, y, z]            Accelerometro
VectorFloat gravity;    // [x, y, z]            GIROSCOPIO
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and
gravity vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0, 0, 0, 0, 0, 0, 0, 0x00, 0x00,
'\r', '\n' };


float senszpiu = -0.015; //sensibilità motori
float sensxpiu = 0.01; //sensibilità motori

float senszmeno = -0.005; //sensibilità motori
float sensxmeno = -0.1; //sensibilità motori

// ================================================================
// ===               INTERRUPT DETECTION ROUTINE                ===
// ================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt
pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}
```

```
//dichiarazione variabili


// ================================================================
// ===                      SETUP                               ===
// ================================================================

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  // inizializzo la comunicazione seriale
  Serial.begin(115200);
  while (!Serial); // wait for Leonardo enumeration, others continue
immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3v or Ardunio
  // Pro Mini running at 3.3v, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.

  // inizializzo il dispositivo
  Serial.println(F("Inizializzazione MPU6050..."));
  mpu.initialize();

  // verifica di connessione
  Serial.println(F("Prova di comunicazione con MPU6050..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connessione riuscita") :
F("MPU6050 Non connesso, controllare i collegamenti e riprovare"));

  // attendo l'utente
  Serial.println(F("\nImmetti un carattere per iniziare.. "));
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available());                 // wait for data
  while (Serial.available() && Serial.read()); // empty buffer again

  // carica dati dal gy521
  Serial.println(F("Programma in esecuzione..."));
  devStatus = mpu.dmpInitialize();

  // regolazione sensibilità e offset giroscopio
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1788 fattore di default del chip

  // test di funzionamento del dispositivo
  if (devStatus == 0) {
    // turn on the DMP, now that it's ready
    Serial.println(F("Avvio del MPU6050..."));
    mpu.setDMPEnabled(true);
    //------------------------------------------------------------------------
-----------------------   COS'è?
    // enable Arduino interrupt detection
    Serial.println(F("Enabling interrupt detection (Arduino external
interrupt 0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
```

```
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to
use it
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;
    //------------------------------------------------------------------------
--------------------------  FINE COS'è?
    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    // ERRORE
    // 1 = overflow di memoria
    // 2 = configurazione mpu6050 fallita
    Serial.print(F("Inizializzazione con MPU6050 fallita (errore "));
    Serial.print(devStatus);
    Serial.println(F(")"));
    void regolamotori(); //definizione della funzione
  }



  // variabili

  pinMode(13, OUTPUT); //motore in basso a destra
  pinMode(12, OUTPUT); //morore in basso a sinistra
  pinMode(11, OUTPUT); //motore in alto a destra
  pinMode(10, OUTPUT); //motore in alto a sinistra




}



// ================================================================
// ===                      MAIN PROGRAM LOOP                   ===
// ================================================================

void loop() {

  // se l'inizializzazione ha fallito non fare niente
  if (!dmpReady) return;

  //------------------------------------------------------------------------
------------------
  while (!mpuInterrupt && fifoCount < packetSize) {
//|
  }
//|
  mpuInterrupt = false;
//|
  mpuIntStatus = mpu.getIntStatus();
//|
  fifoCount = mpu.getFIFOCount();
//|
  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
//|
    mpu.resetFIFO();
//|
    Serial.println(F("contatore fifo in overflow riavviare programma!"));
//|
  } else if (mpuIntStatus & 0x02) {
```

```
//|
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
//|
    mpu.getFIFOBytes(fifoBuffer, packetSize);
//|
    fifoCount -= packetSize;
//| parte di acquisizione dei dati del
#ifdef OUTPUT_READABLE_YAWPITCHROLL
//| giroscopio
    mpu.dmpGetQuaternion(&q, fifoBuffer);
//|
    mpu.dmpGetGravity(&gravity, &q);
//|
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
//|
    //print dei dati dell'mpu6050
//|
    Serial.print("mpu6050 \t");
//|
    //giroscopio (da noi non utilizzato)
//|
    //   Serial.print(ypr[0] * 180/M_PI); //giroscopio
//|
    //   Serial.print("\t");
//|
    //accellerometro
    Serial.print("X= ");          //|
    Serial.print(ypr[1] * 180 / M_PI); //asse x
//|
    Serial.print("\t");
//|
    Serial.print("Z= ");
//|
    Serial.println(ypr[2] * 180 / M_PI); //asse z
//|
#endif              //|
    //* 180/M_PI
    //-------------------------------------------------------------------------
--------------------


    if (ypr[1] > sensxpiu || ypr[1] < sensxmeno && ypr[2] > senszpiu ||
ypr[2] < senszmeno )
    {
      Serial.print("stabilizza:  ");
      regolamotori();
    }
    else {
      digitalWrite(13, LOW);
      digitalWrite(12, LOW);
      digitalWrite(11, LOW);
      digitalWrite(10, LOW);




    }


  }
}


void regolamotori()
```

```
{
  //------MOTORE 1
  if (ypr[1] > sensxpiu && ypr[2] > senszpiu)
  {
    //aziona motore a
    digitalWrite(13, HIGH);
    Serial.print("motore 1");
  }
  else
  {
    digitalWrite(13, LOW);
  }
  //------MOTORE 2
  if (ypr[1] < sensxmeno && ypr[2] > senszpiu)
  {
    //aziona motore b
    digitalWrite(12, HIGH);
    Serial.print("motore 2");
  }
  else
  {
    digitalWrite(12, LOW);
  }
  //-------MOTORE 3
  if (ypr[1] > sensxpiu && ypr[2] < senszmeno )
  {
    //aziona motore d
    digitalWrite(11, HIGH);
    Serial.print("motore 3");
  }
  else
  {
    digitalWrite(11, LOW);
  }
  //-------MOTORE 4
  if (ypr[1] < sensxmeno && ypr[2] < senszmeno)
  {
    //aziona motore c
    digitalWrite(10, HIGH);
    Serial.print("motore 4");
  }
  else
  {
    digitalWrite(10, LOW);
  }
  //-------due motori insieme
  //-------se 1 e 3 sono sballati
  if (ypr[1] > sensxpiu && ypr[2] > senszpiu && ypr[1] > sensxpiu && ypr[2] <
senszmeno)
  {
    //aziona motore 1 e 3
    digitalWrite(11, HIGH);
    digitalWrite(13, HIGH);
    Serial.print("motore 1 e 3");
  }
  else {
    digitalWrite(11, LOW);
    digitalWrite(13, LOW);
  }

  byte inserimento = Serial.read(); //LEGGO IL 1 BYTE
  return;
}
```

Link Utili

www.elettronica.yolasite.com
www.robertagerboni.it
www.arduino.cc
www.wikipedia.com/it