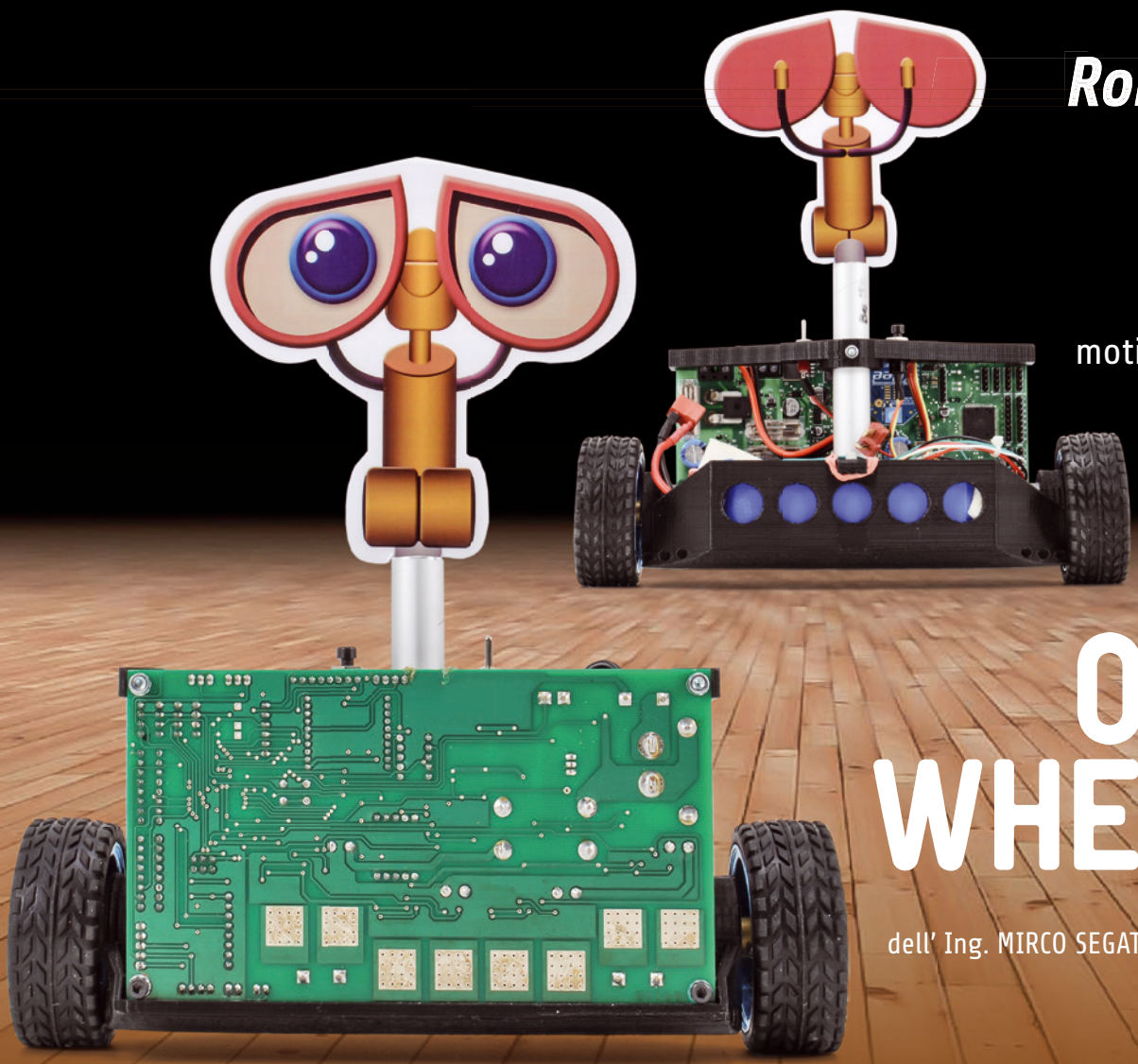


Scheda scalabile per prototipare applicazioni di motion control sia di balancing robot sia di qualsiasi apparato su ruote.

OPEN WHEELS 2.0

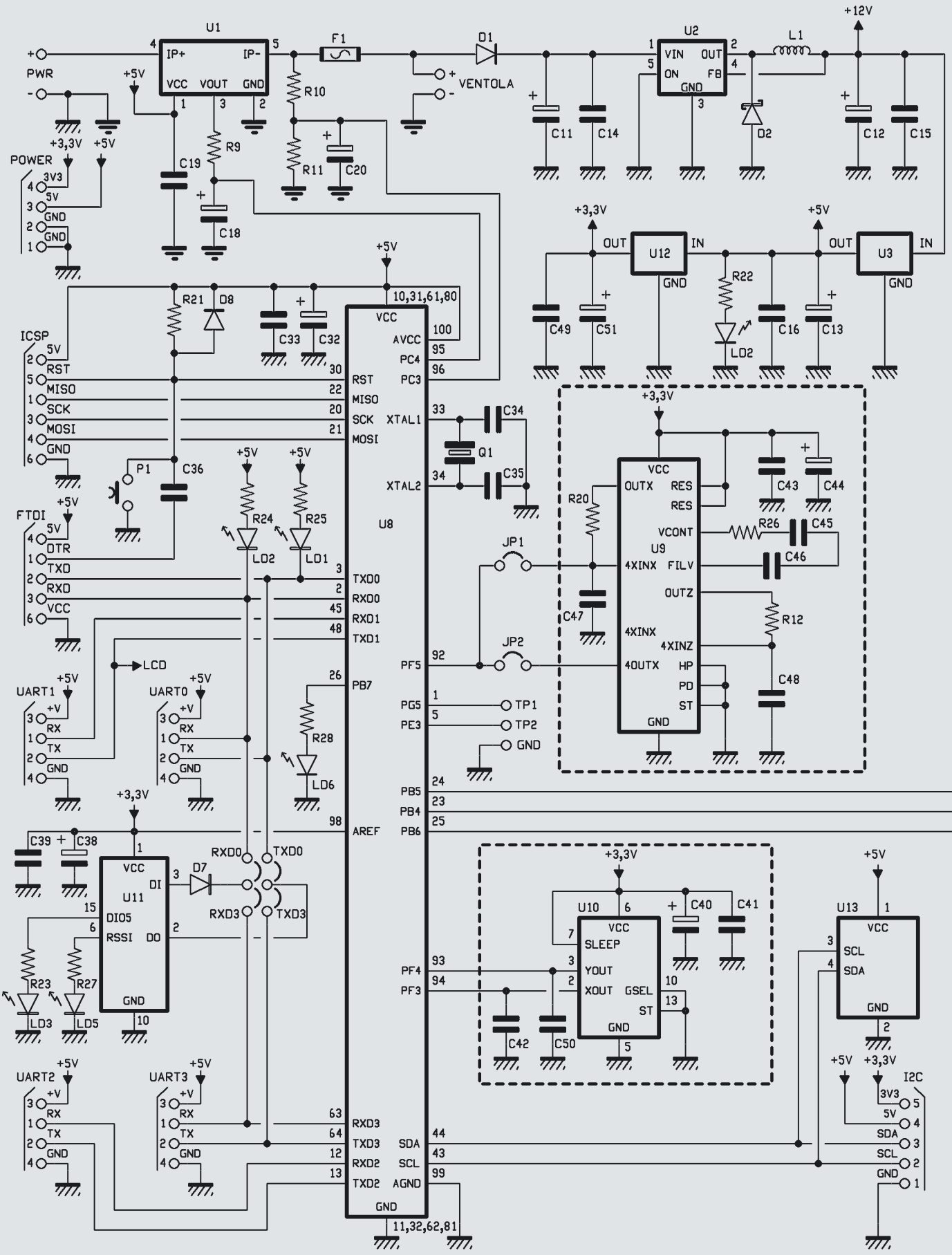
dell' Ing. MIRCO SEGATELLO

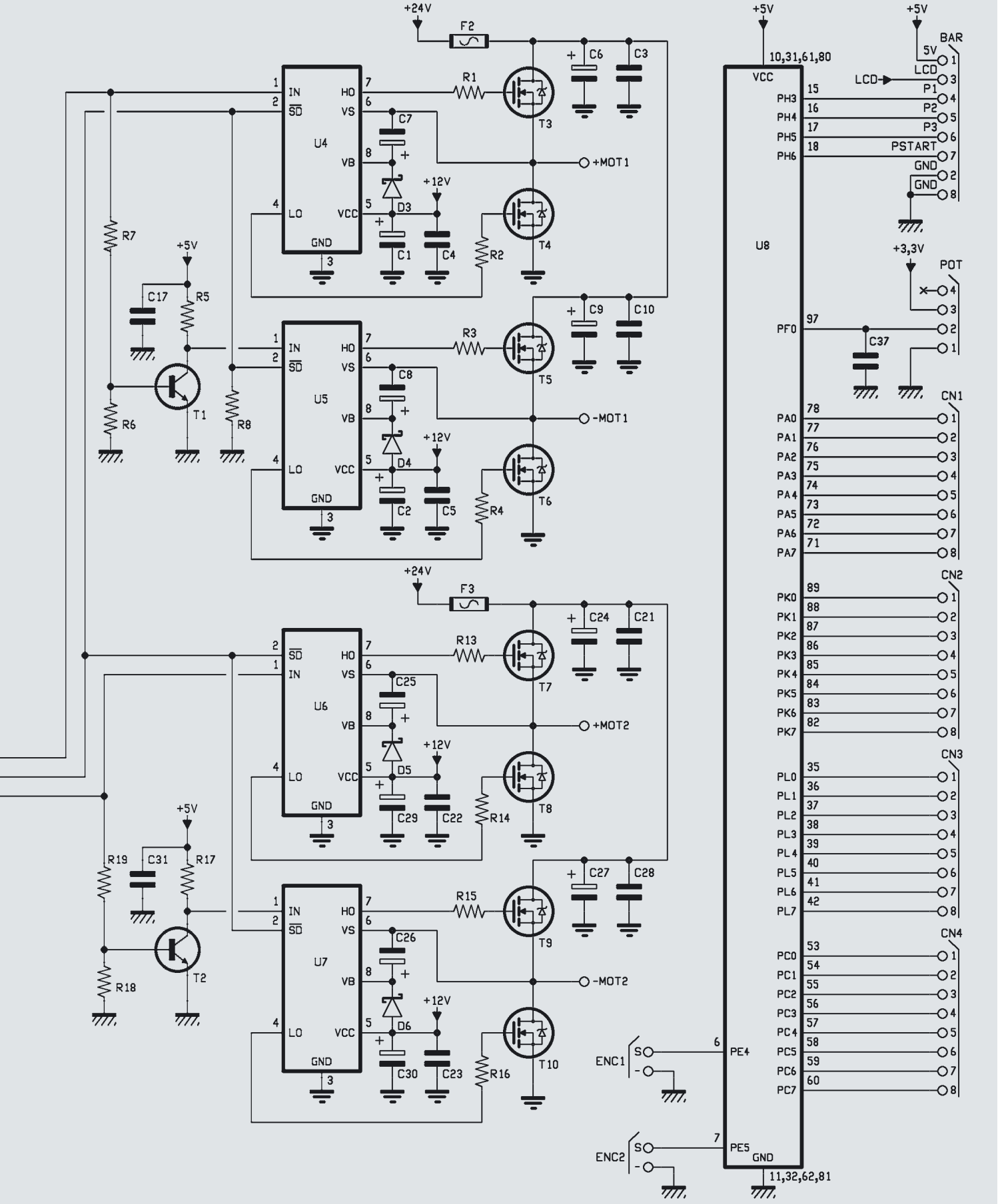
Era il 2013 quando presentammo una scheda di controllo per realizzare un clone del famoso SegWay: era il progetto OpenWheels. In questi ultimi anni l'attenzione verso i sistemi che permettono di stabilizzare un sistema intrinsecamente instabile non si è attenuata, anzi, basta guardare i numerosi progetti proposti in rete per vedere quanta gente si sia cimentata nel realizzare semplici (e non) robot equilibristi, meglio noti col termine anglosassone "balanced robot". Dal canto nostro, abbiamo evoluto il nostro primo progetto mettendo a frutto le nuove tecnologie che il mercato ci mette a disposizione; così è nata una nuova scheda che abbiamo chiamato OpenWheels2, la quale consiste in una board all-in-one utile a realizzare ogni vostro progetto rivolto alla movimentazione di robot. La scheda che andiamo a presentare è infatti equipaggiata (ed espandibile) con tutto l'occorrente per realizzare robot anche complessi, quindi non solo del tipo a bilanciamento. Il progetto è stato dirottato dal controllo di un clone di SegWay verso un robot bilanciato che chiameremo Personal Robot.

CARATTERISTICHE

La principale innovazione riguarda il processore, che passa da un ATmega328P ad un ATmega2560 mantenendo la compatibilità con Arduino e aumentando considerevolmente i pin disponibili e i moduli UART, molto utili per le funzioni di connettività. Il modulo IMU è ora più prestante, mentre gli stadi di alimentazione e a MOSFET sono rimasti invariati. La nuova scheda ha dimensioni simili a quelle della precedente e i connettori sono invariati, un facile aggiornamento a quanti abbiano realizzato OpenWheels.
Le caratteristiche della nuova scheda sono:

- compatibile con Arduino 2560;
- predisposta per utilizzare moduli XBee e Bluetooth;
- supporta sino a due motori DC con corrente massima di 20A (30A di picco);
- dispone di sensori di tensione e corrente sull'alimentazione;
- predisposta con ingresso per encoder;
- supporta moduli IMU sia analogici che digitali.


U1
IP+
IP-
VCC
VOUT
GND
PWR
+5V
+3,3V
POWER
3V3
5V
GND
F1
D1
VENTOLA
R10
R11
R9
C19
C18
C20
U2
VIN
OUT
ON
FB
GND
L1
D2
+12V
C11
C14
C12
C15
U12
U3
IN
OUT
C49
C51
C16
C13
R22
LD2
ICSP
RST
MISO
SCK
MOSI
R21
D8
C33
C32
C36
P1
FTDI
DTR
TXD
RXD
R24
R25
LD1
LD2
LCD
UART0
UART1
UART2
UART3
R28
LD6
C39
C38
U11
DIO5
RSSI
DO
D7
R23
R27
LD3
LD5
U8
VCC
AVCC
PC4
PC3
XTAL1
XTAL2
C34
C35
Q1
TXD0
RXD0
RXD1
TXD1
PB7
AREF
PF5
PG5
PE3
JP1
JP2
TP1
TP2
PB5
PB4
PB6
PF4
PF3
RXD3
TXD3
RXD2
TXD2
SDA
SCL
AGND
11,32,62,81
U9
OUTX
RES
VCONT
4XINX
FILV
OUTZ
4XINZ
4OUTX
HP
PD
ST
R20
R26
C45
C46
C47
C48
C43
C44
R12
U10
SLEEP
YOUT
XOUT
GSEL
C40
C41
C42
C50
U13
I2C


+24V
F2
C6
C3
R1
T3
U4
IN
SD
HO
VS
VB
LO
VCC
GND
C7
D3
+12V
C1
C4
R2
T4
+MOT1
R7
+5V
C17
R5
T1
R6
R8
C9
C10
R3
T5
U5
C8
D4
C2
C5
R4
T6
-MOT1
F3
C24
C21
R13
T7
U6
C25
D5
C29
C22
R14
T8
+MOT2
R19
C31
R17
T2
R18
C27
C28
R15
T9
U7
C26
D6
C30
C23
R16
T10
-MOT2
+5V
10,31,61,80
VCC
U8
PH3
PH4
PH5
PH6
15
16
17
18
BAR
5V
LCD
P1
P2
P3
PSTART
GND
+3,3V
POT
PF0
97
C37
CN1
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
78
77
76
75
74
73
72
71
CN2
PK0
PK1
PK2
PK3
PK4
PK5
PK6
PK7
89
88
87
86
85
84
83
82
CN3
PL0
PL1
PL2
PL3
PL4
PL5
PL6
PL7
35
36
37
38
39
40
41
42
CN4
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
53
54
55
56
57
58
59
60
ENC1
ENC2
S
PE4
PE5
6
7
11,32,62,81

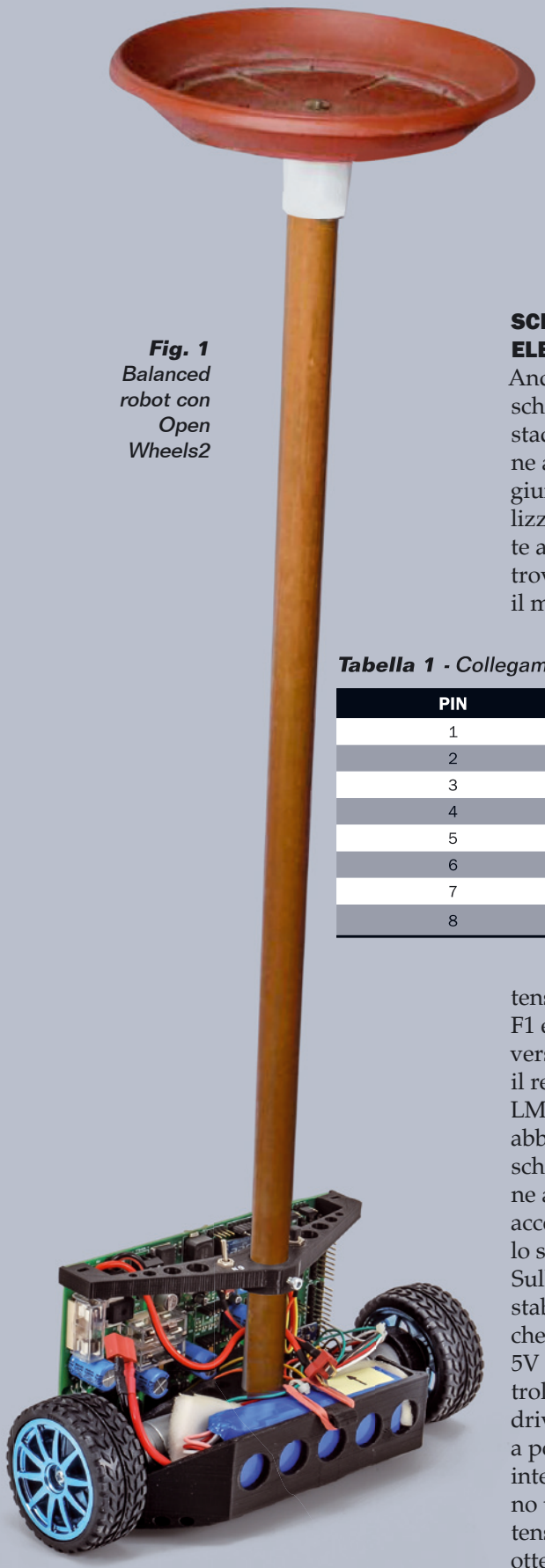


Fig. 1 *Balanced robot con Open Wheels2*

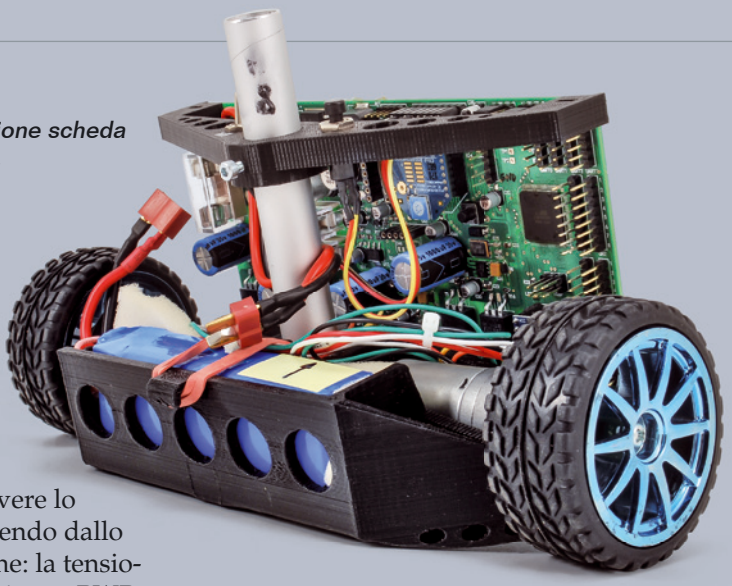


Fig. 2 *Installazione scheda e motori.*

SCHEMA ELETTRICO

Andiamo ora a descrivere lo schema elettrico, partendo dallo stadio di alimentazione: la tensione applicata ai contatti + e - PWR giunge all'integrato ACS758, utilizzato per la misura della corrente assorbita, all'uscita del quale si trovano il partitore R10/R11 che il micro utilizza per misurare la tensione della batteria, il fusibile F1 e il diodo di protezione dall'inversione di polarità, che precede il regolatore di tensione switching LM2576 (U2) utilizzato per abbassare la tensione a 12V. La scheda nasce per un'alimentazione a 24V e, senza alcuna modifica, accetta fino a 11,5V; bypassando lo switching è possibile scendere. Sulla linea dei 12V troviamo due stabilizzatori di tensione lineari che forniscono rispettivamente i 5V e i 3,3V necessari al microcontrollore e al modulo IMU. I due driver motori, in configurazione a ponte H, sono realizzati con integrati IRS2184, i quali sfruttano un sistema di elevamento di tensione a pompa capacitiva per ottenere una tensione maggiore di quella dell'alimentazione, così da pilotare il gate dei MOSFET. Per ciascun motore vengono utilizzati due integrati IRS2184 e quattro MOSFET pilotati con soli due segnali: uno è il segnale che abilita il driver (shutdown) mentre il secondo è il segnale PWM di comando configurato in modalità locked-antiphase. In questa modalità, il segnale PWM giunge a entrambi i rami del ponte H ma in opposizione di fase: così la ruota rimane ferma con un duty-cycle del 50%. Ruoterebbe alla massima velocità in una direzione con un duty-cycle di 0% e alla massima velocità in senso opposto con duty-cycle del 100%. Nel caso di balanced-robot in cui le ruote presentano continui cambi di direzione (per mantenere la posizione verticale); tale modalità permette di far lavorare i MOSFET di potenza con un duty-cycle attorno al 50%, evitando segnali impulsivi molto rapidi come accade nella classica mo-

Tabella 1 - *Collegamenti del connettore BAR.*

PIN	Funzione
1	+5V
2	GND
3	LCD
4	P1 (-)
5	P2 (+)
6	P3 (E)
7	PSTART
8	GND

Tabella 2 - *Collegamenti potenziometro sterzo connettore POT.*

PIN	Funzione
1	GND
2	POT
3	+5V
4	+3V3

dalità sign-magnitude, in cui un segnale è adibito alla direzione ed uno al PWM. In questi ultimi driver, quando il robot si trova nella posizione di equilibrio il segnale PWM ha un duty-cycle molto basso ed il segnale di direzione ha continui cambiamenti di stato; se per motori che assorbono poche centinaia di milliampere ciò non costituisce un problema, nel caso di motori che assorbono diversi ampere può diventarlo. I MOSFET previsti nel progetto sono del tipo AUIRF2804, capaci di gestire correnti impulsive di 200A e correnti continue di almeno 10A, senza riscaldare apprezzabilmente; abbiamo comunque previsto un connettore cui connettere una ventola per il raffreddamento forzato. Tre, sono i fusibili a protezione della scheda: uno da 1A per i circuiti di comando e due sulle due linee dei motori; questi ultimi vanno scelti in modo da interrompere il circuito di potenza in presenza di picchi di corrente eccessivi, solitamente presenti al momento dell'alimentazione da fermo dei motori. Il microcontrollore è un ATmega2560, lo stesso montato sulla scheda Arduino Mega, che, rispetto all'ATmega328 (della Arduino UNO), dispone di un maggiore numero di pin e di ben quattro moduli UART per la comunicazione seriale hardware. Gli ingressi analogici sono utilizzati per la lettura della corrente e della tensione di alimentazione, oltre che del potenziometro dello sterzo (se presente) e dell'accelerometro e del giroscopio analogici, qualora siano presenti. È previsto un connettore di nome BAR, cui connettere dei pulsanti e un eventuale display LCD secondo la **Tabella 1**, ed un connettore per il potenziometro dello sterzo (**Tabella 2**).
Sulla scheda sono presenti i socket per l'accelerometro analogico MMA7361 e per il giroscopio analogico LPR403, oltre a quello per il modulo XBee; quattro connettori rendono disponibili i segnali TX ed RX degli altrettanti UART del microcontrollore, con relativa alimentazione. La seriale principale Serial0 fa capo al connettore FDTI, al quale può essere connesso un convertitore USB/seriale utile alla programmazione del microcontrollore, mentre la seriale Serial3 viene utilizzata per la telemetria e connessa al socket per il modulo XBee tramite i jumper XB1TX e XB1RX; la seriale Serial1 viene invece usata per il display LCD. Nella parte inferiore della scheda troviamo il connettore per il modulo IMU digitale MPU-6050 della InvenSense e quello per la porta di comunicazione I²C-Bus, al quale collegare qualsiasi altro modulo IMU digitale. È presente anche un connettore per l'ingresso dei segnali provenienti da due encoder ed uno per l'alimentazione generica di moduli esterni. Tutti i pin dell'ATmega2560 non utilizzati nel progetto sono comunque resi disponibili sui connettori CN1,CN2,CN3 e CN4, utili per ogni eventuale espansione che desideriate implementare. Per ultimo abbiamo il connettore ICSP, che, come vedremo, sarà utilizzato per caricare il bootloader sul microcontrollore.
Sono stati implementati diversi LED per segnalare lo stato di funzionamento, come riportato nella **Tabella 3**.
Conclusa la descrizione dello schema, passiamo all'utilizzo pratico della scheda cimentandoci nella realizzazione di un balance-robot particolarmente sofisticato, in grado di mantenersi perfettamente bilanciato anche nelle situazioni più critiche.

***Tabella 3** - Funzionamento dei LED*

LED	Funzione
LD1	UART0 TX (USB/seriale monitor)
LD2	Presenza 5V
LD3	XBee ASS
LD4	UART1 RX (USB/serial monitor)
LD5	XBee RSSI
LD6	Connesso al pin 13 di Arduino

LA MECCANICA DEL ROBOT

Partiamo dalla meccanica, stampata con la nostra 3Drag ed opportunamente disegnata per accogliere l'elettronica (i file STL li potete scaricare dal nostro sito

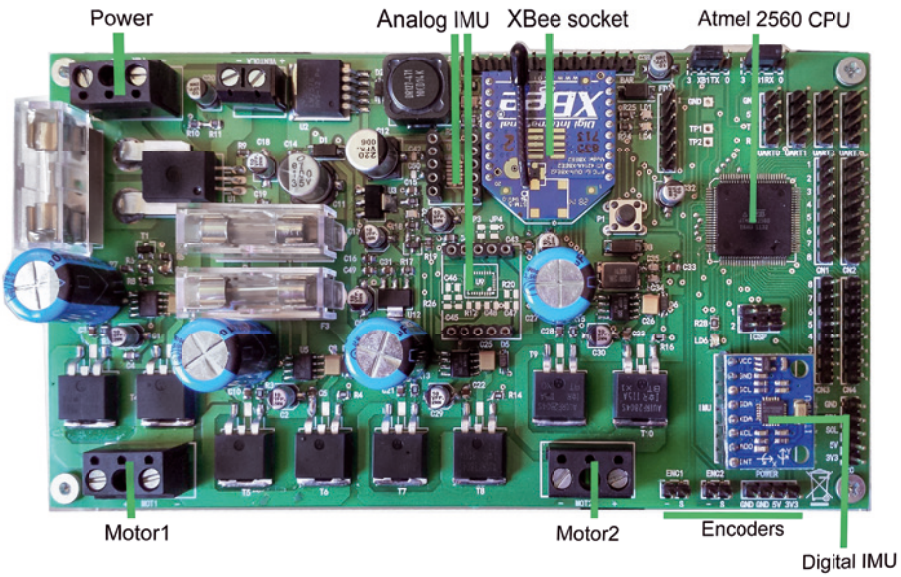


***Fig. 3** - Connettori della scheda di controllo.*

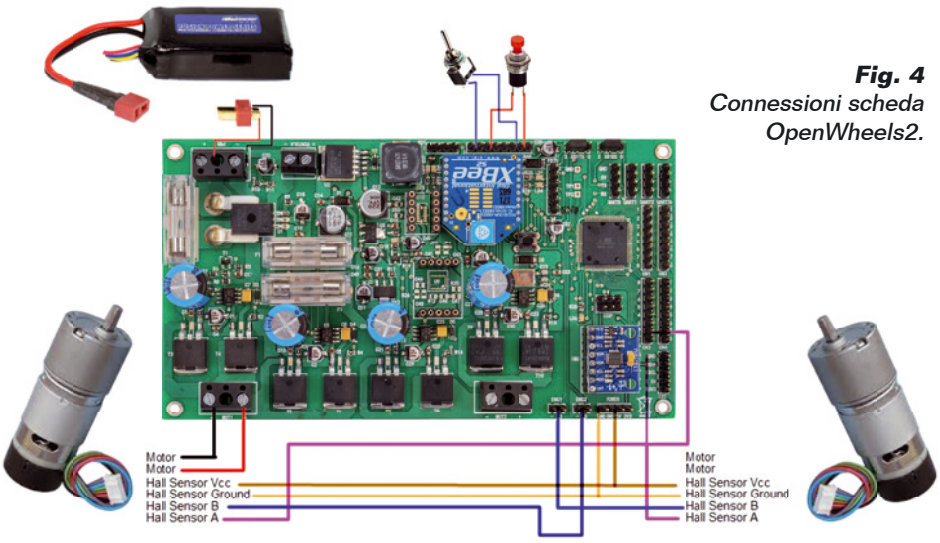


Fig. 4
Connessioni scheda OpenWheels2.

www.elettronicain.it). Su questo basamento andremo a fissare i motori, la batteria e la scheda OpenWheels2. Sulla base fisseremo un'asta ricavata da un tubo di alluminio, sulla cui sommità abbiamo posto per le prove un piccolo vassoio che ci permetterà di contenere anche degli oggetti o, come abbiamo fatto noi, dei drink; nulla vieta di applicare qualsiasi altra cosa.

Alla scheda di controllo vanno collegati i seguenti dispositivi:

- 2 motoriduttori con encoder e ruote (2846-BALANCINGBOT);
- batteria LiPo 1.800 mAh/11,1 V (5400-LIPO1800);
- connettore maschio/femmina (7300-TCONNMF);
- deviatore unipolare a pulsante (8220-8701);
- deviatore unipolare a levetta per c.s. a 90° (8220-TS-8);
- 2 moduli Xbee 2 mW serie 2 (6168-XBEE2MW);
- XBEE EXPLORER USB (6168-XBEEDONGLE2);
- modulo Bluetooth controller DAGU (2586-DG010);
- accelerometro 3 assi + Giroscopio a 3 assi (2846-MPU6050);
- convertitore FTDI USB/seriale 3,3V e 5V (7300-FTDI5V);
- fusibile (F2) 1A;
- fusibile (F1,F3) 5A.

I moduli XBee e Bluetooth servono per la telemetria. I componenti si trovano tutti presso Futura Elettronica (*www.futurashop.it*) e tra parentesi trovate i rispettivi codici.

Nel caso si utilizzasse un'asta lunga, potrebbe essere necessario prevedere un piccolo peso di circa 100 g per bilanciare il peso alla base dell'asta, dovuto ai motori ed alla batteria; quest'ultima avremmo potuto posizionarla in alto, ma avrebbe richiesto lunghi cavi che avrebbero introdotto una resistenza elettrica deleteria.

Anche se non sembra, il peso posto in alto migliora la stabilità minimizzando nel contempo piccole oscillazioni dovute alla cedevolezza dei materiali.

A questo punto fissate i motori al supporto stampato in 3D, le relative ruote e la scheda elettronica, come visibile nelle foto dell'articolo; l'unica precauzione è fissare a dovere la scheda. Per quanto riguarda i collegamenti elettrici, occorre fare molta attenzione, soprattutto per i motori che hanno incorporato l'encoder; riferitevi alla **Fig.3**.

È previsto un interruttore che sarà utilizzato per armare il robot connesso secondo le indicazioni della **Tabella 4** ed un pulsante per il reset degli allarmi. Se volete, potete inserire un interruttore di accensione in serie alla batteria. La **Fig. 4** illustra il cablaggio.

IL FIRMWARE

Il firmware è derivato da quello di OpenWheels, migliorato ed arricchito di nuove funzionalità. Per caricarlo è necessario installare il bootloader nell'ATmega; allo scopo possiamo usare un apposito programmatore, oppure una comune scheda Arduino usata come programmatore.

In quest'ultimo caso aprite l'IDE di Arduino e caricate lo sketch di esempio denominato *ArduinoISP.ino* sulla scheda Arduino, quindi impostate come scheda target "Arduino Mega2560 or Mega ADK" accedendo al menu Strumenti>tipo di Arduino, poi impostate Arduino come programmatore (Strumenti>Programmatore>Arduino as ISP).

A questo punto avviate la programmazione con il comando Strumenti>Scrivi il Bootloader.

A procedura completata (sono richiesti alcuni minuti) sconnettete il programmatore ed interfacciatevi direttamente al PC tramite il connettore FDTI con l'ausilio di un convertitore USB/Seriale: la vostra scheda sarà vista dall'IDE come una Arduino Mega.

Per verificare che tutto sia funzionante, caricate lo sketch di esempio *Blink.ino* e accertatevi che il LED LD6 lampeggi.

Per il momento potete omettere il modulo XBee ed assicurarvi che il modulo IMU sia correttamente installato; potete quindi fare alcuni test per verificare che tutto funzioni (consigliamo di farlo prima di installare la scheda ed effettuare le connessioni).

Testiamo i sensori caricando sulla scheda lo sketch di nome *PersonalRobot_TestSensor.ino* disponibile con i file del progetto su *www.elettronicain.it*: i dati

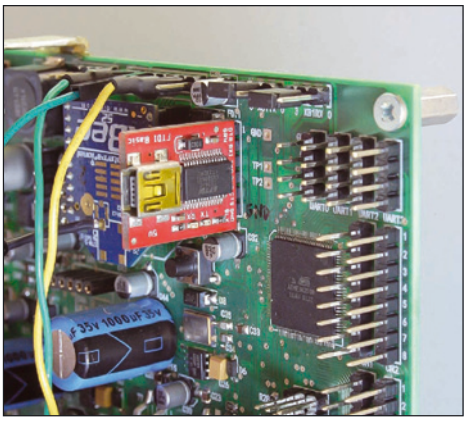
Tabella 4 - *Collegamenti elettrici motore sinistro*

Componente	pin usati
Interruttore stand-by/Go	Connesso tra i pin 2 e 7 del connettore BAR
Pulsante di reset (NA)	Connesso tra i pin 4 e 8 del connettore BAR

Tabella 5 - *Collegamenti per caricare il bootloader usando una scheda Arduino come programmatore.*

Arduino (programmatore)	Openwheel2 (da programmare)
Arduino pin D12/MISO	ICSP pin 1
Arduino pin +5V	ICSP pin 2
Arduino pin D13/SCK	ICSP pin 3
Arduino pin D11/MOSI	ICSP pin 4
Arduino pin D10/RESET	ICSP pin 5
Arduino pin GND	ICSP pin 6

Fig. 5 - *Montaggio del convertitore USB/seriale.*



acquisiti saranno visualizzati direttamente su SerialMonitor di Arduino in modo grezzo, senza alcuna post-elaborazione. Con un semplice jumper potete simulare i pulsanti cortocircuitando gli ingressi P1, P2, P3, PStart verso il pin GND; connettendo un display seriale sui pin +5V, GND, LCD del connettore BAR potrete verificarne il funzionamento. Ricordiamo che il display LCD è supportato sia nell'hardware che nel software, ma non utilizzato in questo progetto in quanto, come vedremo, abbiamo previsto altri sistemi per la visualizzazione dei dati. Se sono collegati i motori, è possibile testarli caricando lo sketch *OpenWheelsTestMotorV2.ino;* il comando dei motori avviene da SerialMonitor con il semplice invio di caratteri. La cosa importante è che i motori ruotino nella direzione corretta e che il robot avanzi quando le ruote girano in senso orario.

Una delle più importanti innovazioni rispetto alla precedente versione è l'utilizzo di un modulo IMU digitale basato sull'integrato MPU-6050 della Invensense, che contiene un accelerometro e un giroscopio entrambi a tre assi, oltre ad un microprocessore idoneo ad elaborare i dati acquisiti. La precisione e la risoluzione sono enormemente aumentati ed il livello di rumore che affligge da sempre gli accelerometri, qui è molto basso. Questo modulo che spesso si trova in commercio come breakout board con la sigla GY-521, è un ottimo compromesso tra prestazioni e costi; inoltre la sua interfaccia di comunicazione I²C è perfetta per l'abbinamento con Arduino. Come sempre la lettura dell'accelerometro e del giroscopio vanno fuse assieme al fine di minimizzare l'inevitabile drift dell'offset del giroscopio e limitare il rumore dell'accelerometro; tra le varie soluzioni, il filtro complementare si è rivelato efficace e facile da implementare (chi volesse approfondire l'argomento troverà interessante questo tutorial in lingua italiana: *http://www.gioblu.com/tutorials/sensori/193-filtro-complementare-e-filtro-di-kalman*).

Il firmware necessario a far funzionare il nostro robot si chiama *PersonalRobot_V10.ino* ed è scaricabile insieme a tutti gli altri file del progetto. Partiamo dicendo che nel progetto la scheda

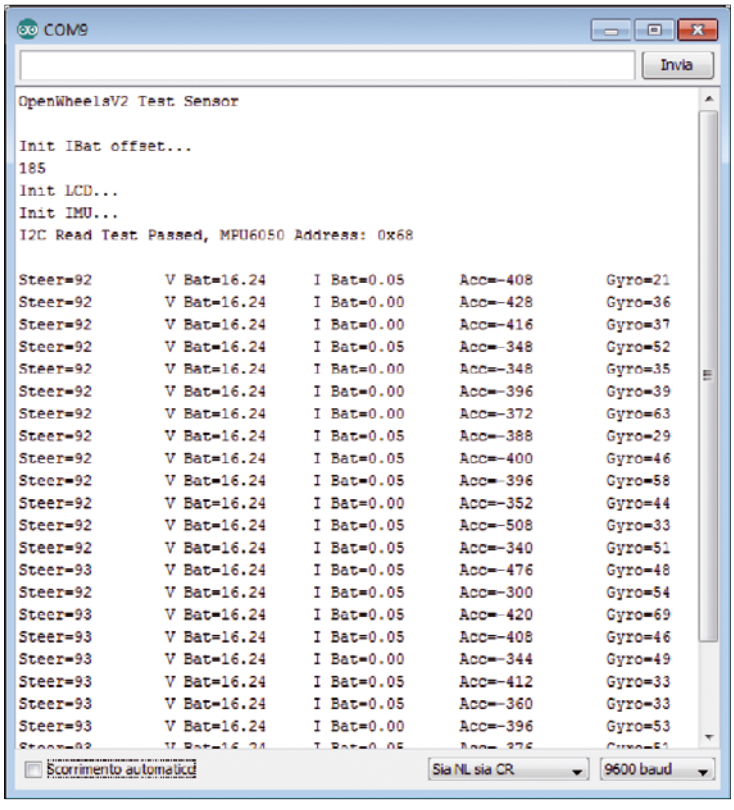


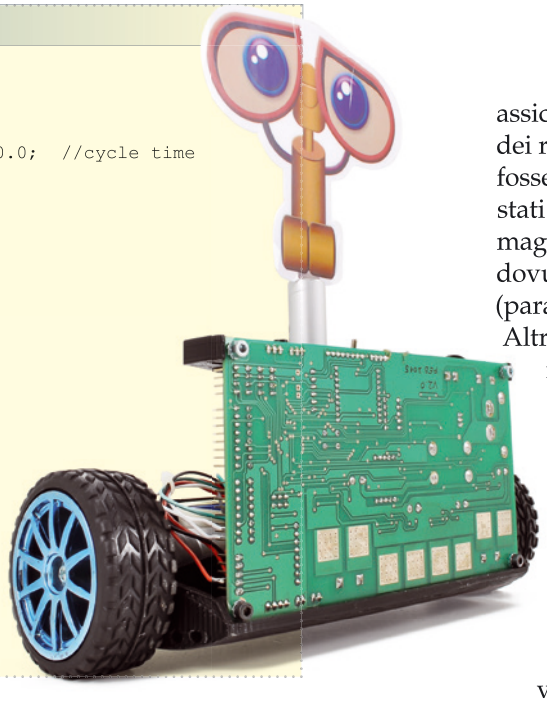
Fig. 6 *Serial monitor con i dati rilevati durante il test della sensoristica.*

Listato 1

```
void loop()
{
  time2=micros();
  dt = float(time2-time1)/1000000.0;  //cycle time
  time1=time2;

  SoftStart();
  EstimateAngle();
  EstimatePos();
  Status();
  PID();
  SetMotor();
  SetZero();
  Telemetry();

  if (digitalRead(P1_pin) == 0)
  {
       // reset error
      statusFlag=0;
      errorIMU=0;
      Integral=0;
      encoderCounterLeft=0;
      encoderCounterRight=0;
  }
  delay(18);  //about 50Hz
}
```



è posta in verticale, ma nulla vieta di posizionarla in orizzontale; in tal caso è necessario configurare il firmware lasciando la sola riga di codice utile:

```
//horizontal board
//Wire.write(MPU6050_ACCEL_YOUT_H);

//vertical board
Wire.write(MPU6050_ACCEL_ZOUT_H);
```

Lo sketch è suddiviso in blocchi, ciascuno dei quali ha un file distinto in base alla funzione svolta; il file *PersonalRobot_V10.ino* contiene il codice principale dal quale sono richiamate le funzioni via-via necessarie.
Il ciclo principale contiene solo le chiamate alle funzioni e la misura del tempo di ciclo (**Listato 1**). L'intero ciclo di stabilizzazione dura circa 2 ms, ma con una funzione delay lo portiamo a un più ragionevole 20 ms, ovvero 50 volte al secondo; la telemetria influenza molto il tempo di ciclo, che comunque viene ricalcolato ad ogni ciclo. Nel file *EEPROM.ino* sono presenti le funzioni necessarie a leggere e scrivere nella memoria EEPROM dell'ATmega2560: nel nostro caso, tutti i parametri del controllo PID ed i valori di offset dei sensori. Nel file *IMU.h* sono contenute le funzioni che permettono di inizializzare i sensori, misurarne l'offset e i valori istantanei. In applicazioni di questo tipo il firmware è molto critico e contempla diversi parametri che devono assumere un ben preciso valore a seconda dell'hardware utilizzato; è sufficiente modificare, ad esempio, l'altezza dell'asta del robot, per modificarne il baricentro ed il momento d'inerzia; immaginate cosa può accadere sostituendo ad esempio i motori o variando qualche peso.
Oltre ai parametri canonici del controllo PID, ce ne sono alcuni meno critici ma ugualmente importanti, che fanno la differenza tra un robot che sta in verticale ed uno che rimane in verticale anche se gli si appoggia un peso all'improvviso; in particolare, nella funzione **MPU6050_init()** osserviamo la riga di impostazione del modulo **MPU6050_write(MPU6050_CONFIG, 0x04)** che imposta un'azione di filtro dei dati dell'accelerometro e del giroscopio alla frequenza di 20 Hz (parametro DLPF_CFG=4) assicurando una pulizia, in lettura, dei rumori indesiderati. Se il robot fosse stato più piccolo e fossero stati necessari tempi di reazione maggiori, questo valore avrebbe dovuto necessariamente salire (parametro DLPF_CFG più basso).
Altro parametro relativamente importante è la frequenza di taglio del filtro complementare che fonde le letture dei sensori. La riga in cui viene definito è questa: float Comp_Filter_Constant = 0.01; valori più bassi rallentano la misura dell'angolo (ma attenuano di più il rumore) mentre valori più alti permettono di ottenere una misura angolare più veloce ma maggiormente affetta da rumore.
La sezione che gestisce gli encoder è contenuta nel file *Encoder.ino*, sezione implementata ex-novo proprio per OpenWheels2: se si realizza un balanced robot, diventa importante non solo la stabilità verticale, ma anche il mantenimento di un preciso punto rispetto al suolo, altrimenti può capitare che il robot se ne vada in giro per la stanza senza controllo a causa del drift di qualche sensore. Gli encoder forniscono informazioni circa la rotazione delle ruote, quindi è possibile valutare quando il robot si è mosso. Utilizzando gli encoder su entrambe le ruote possiamo anche avere informazioni sulla differenza di rotazione tra le due, cosa che porterebbe il robot a ruotare su se stesso a causa delle differenze tra i motori. Per questo motivo vengono determinate le variabili **WheelsPos (DEG)**, che fornisce la posizione delle ruote in gradi, **WheelsVel**, che fornisce la velocità di rotazione delle ruote in gradi al secondo, e la **encoderDif**, che fornisce indicazione sulla differenza di rotazione tra le ruote in gradi.

A ciascuno dei due encoder è associato un interrupt:

```
attachInterrupt(0, EncoderLeft,
FALLING); //encoder pin2 int0 (ENC1)
attachInterrupt(1, EncoderRight,
FALLING); //encoder pin3 int1 (ENC2)
```

Per sapere se la ruota sta girando in senso orario (e quindi incrementare il conteggio) oppure in senso antiorario (per decrementare il conteggio) dobbiamo sfruttare entrambi i segnali in quadratura forniti dagli encoder. Il secondo segnale è quindi connesso ad un ingresso digitale della scheda e la sua lettura durante l'interruzione fornisce indicazioni sul verso di rotazione della ruota (vedere **Listato 2**).
Il file denominato *PID.ino* è il cuore di tutto il sistema, perché contiene le righe di codice necessarie alla stabilizzazione del robot (**Listato 3**). Si tratta, in definitiva, di un calcolo che tiene conto dei vari contributi in base a parametri che l'utente dovrà inserire in modo molto preciso. Vediamone il significato:

- Pval, controllato dal parametro KP, è il valore proporzionale (più il robot si inclina e più è necessario compensare);
- Ival, controllato dal parametro KI, è il valore integrativo e viene usato solo in cloni di SegWay e controlla la funzione per cui ad andatura costante è possibile riportare il manubrio in posizione verticale;
- Dval, controllato dal parametro KI, è il valore derivativo e fa sentire il suo effetto tanto quanto maggiore è la tendenza a cadere;
- Ppos, controllato dal parametro KO, è il valore proporzionale per il posizionamento; maggiore è questo valore e maggiore sarà la precisione con la quale il robot mantiene la posizione rispetto al suolo;
- Pvel, controllato dal parametro KV, è il valore che insieme al valore Ppos permette di evitare che il robot si allontani dalla posizione iniziale.

Nel software di telemetria c'è la possibilità di impostare il parametro STEER, utile per impostare la sensibilità dello sterzo nel caso il progetto riguardi il clone di un Segway. Un ulteriore parametro denominato Kdm permette di mantenere l'orientamento originale del robot compensando eventuali differenze nella rotazione delle ruote. Ovviamente gli ultimi tre valori devono essere azzerati in tutti i casi in cui si vuole che il robot si sposti, magari perché comandato a distanza.
Analizzando il codice troviamo anche una riga utilizzata per compensare il livello di carica della batteria: drive = drive * VBatAlim/VBatLevel; infatti a parità di potenza calcolata, quella effettiva sulle ruote, dipenderà dal livello di carica della batteria e noi non volgiamo che il robot si comporti in modo differente a seconda del livello della batteria.
Il file *Motor.ino* (**Listato 4**) contiene le impostazioni e le funzioni per gestire i motori, primo fra tutti l'impostazione dei registri interni dell'Atmel, al fine di comandare i motori con un segnale PWM ad alta risoluzione (10bit) invece dei canonici 8bit usati per default in Arduino; inoltre la frequenza del PWM viene portata a valori di 16 kHz, assicurando un movimento più fluido e silenzioso dei motori. Sempre in questa sezione, troviamo altri due parametri denominati *steerSensibilty* e *speedSensibility* che, come suggerito dai loro nomi, definiscono la sensibilità rispettivamente nella sterzata e nella velocità, qualora il robot venga comandato a distanza (**Listato 5**). In questa sezione troverete anche la compensazione della cosiddetta *motor dead zone*, che è il livello di tensione sotto il quale le ruote non girano. Se ad esempio il robot ha bisogno del 5% di potenza sui motori, ma questa non è sufficiente a vincere gli attriti, le ruote non gireranno, con il risultato che il robot tenderà a cadere e richie-

Listato 2

```
void EncoderRight()
{
  // If pinA and pinB are both high or both low, it is spinning
  // forward. If they're different, it's going backward.
     if(PINL & 0b00000010)
        encoderCounterRight++;
     else
        encoderCounterRight--;
}
```

Listato 3

```
Pval = float(KP) * Angle;
Ival = float(KI) * Integral;
Dval = float(KD) * Gyro_Rate;
Ppos = float(KO) * wheelsPos;
Pvel = float(KV) * wheelsVel;
drive = Pval + Ival + Dval + Ppos + Pvel;
drive = drive * VBatAlim/VBatLevel;  // for compensate Battery level
drive = constrain(drive, -250.0, 250.0);  //limit drive values
```

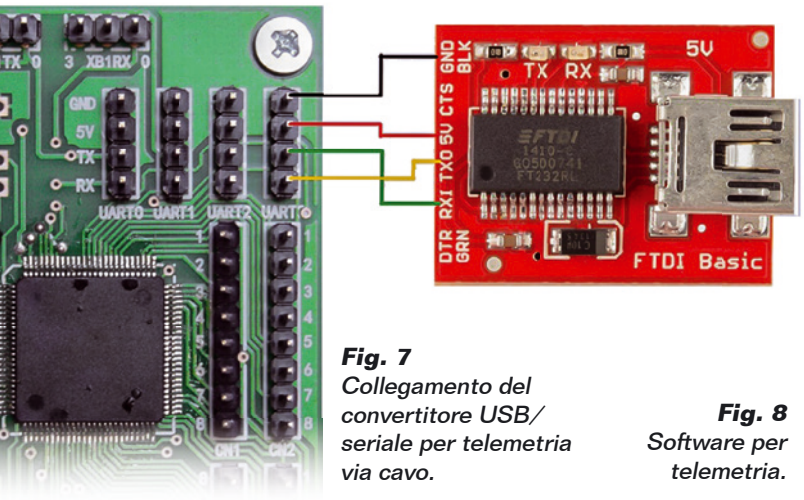


Fig. 7
Collegamento del convertitore USB/ seriale per telemetria via cavo.

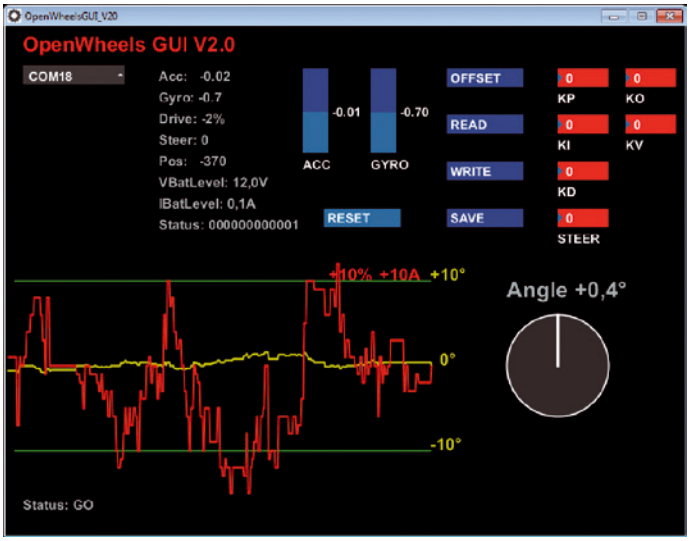


Fig. 8
Software per telemetria.

derà un'ulteriore correzione, con il risultato di un bilanciamento incerto. Una semplice prova con il robot appoggiato al suolo vi permetterà di determinare il minimo valore di potenza alle ruote affinché il robot si muova. La riga di codice che definisce questo parametro è: int deadZoneMotor = 10. La sezione *SerLCD.ino* attualmente non è utilizzata ed è stata lasciata per la compatibilità con il precedente hardware. Il file *telemetry.ino* contiene le funzioni utilizzate per le comunicazioni con la scheda; nello specifico, per richiedere dei dati o impostare dei parametri. In ogni caso è l'apparecchiatura esterna a dover far richiesta di dati, in quanto Openwheels2 non invia in automatico alcun dato; si veda, al riguardo, la **Tabella 6**. Ulteriori parametri di contorno sono usati per controllare lo stato di funzionamento ed eventualmente intervenire con degli allarmi; è il caso dei seguenti tre valori di tensione riferiti rispettivamente al valore massimo, minimo e nominale di alimentazione:

```
float VBatLmax = 12.6; //Max batt Volt
float VBatLmin = 11.5; //Min batt Volt
float VBatAlim = 11.8;  //Nominal batt Volt
```

Tali valori sono usati, oltre che per gli allarmi, anche per la funzione di compensazione di potenza sui motori.

Notare come a causa del driver di potenza la tensione minima di funzionamento non può essere inferiore a 11,5 volt, pena l'impossibilità di portare i MOSFET nello stato di ON. Un ulteriore valore riguarda la massima corrente oltre la quale interviene lo shutdown dei motori; la funzione interviene in circa 20 ms, quindi non ha la stessa rapidità dei fusibili, ma offre una valida protezione nella maggior parte dei casi:

```
float IBatMAX = 3.0;
// IBat MAX for shutdown MOTOR
```

Ulteriori interventi di protezione riguardano le variabili riportate nella **Tabella 7**. Il primo bit della variabile statusFlag non contiene un errore ma indica se il robot è in stato di standby oppure è armato, ovvero attivo e funzionante; questa funzionalità è gestita tramite l'apposito interruttore. Nello stato di standby tutti i sistemi sono pienamente funzionanti, compresa la telemetria, ma il driver è disabilitato e le ruote non possono girare: potete quindi gestire il robot senza la preoccupazione di fare danni.

Listato 4

```
void initMotor()
{
  pinMode(ShutDown_pin, OUTPUT);
  digitalWrite(ShutDown_pin, LOW);   //ShutDown driver at startUP
  pinMode(PWMLeft_pin, OUTPUT);
  pinMode(PWMRight_pin, OUTPUT);

  // Phase and Frequency correct PWM on TIMER1
  //PWM_frequency = (16000000)/(2*Prescale_factor*ICR1)
  //ICR1=(16000000)/(2*Prescale_factor*PWM_frequency)
  // ICR1=400  fPWM=20000
  // ICR1=500  fPWM=16000
  // ICR1=666  fPWM=12012
  // ICR1=1024 fPWM=7812
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) ;
  TCCR1B = _BV(WGM13) | _BV(CS10);
  ICR1=500;

  // ***************************************************************************
  // limit for driver compatibiliti  (5% di ICR1A) < drivePWM < (95% di ICR1A)
  // stop motor driver=stallMotor=ICR1/2;
  // ***************************************************************************
  maxPWM = float(ICR1)*0.80/2; //0.95
  stallMotor=ICR1/2;   //25
  OCR1A = stallMotor;  //0=0%  ICR1/2=50%  ICR1=100%
  OCR1B = stallMotor;
}
```

Il LED LD6 lampeggia durante uno stato di errore; solo premendo il pulsante di reset (o tramite comando remoto) è possibile resettare gli allarmi. Vista la complessità dell'impostazione dei parametri e la necessità di visualizzare lo stato operativo, abbiamo implementato un apposito software (*OpenWheelsGUI_V20*) per la gestione in remoto di tutte le principali operazioni, che permetta di visualizzare tutti i dati di telemetria anche graficamente. Il software è stato scritto con Processing, perché è un ambiente di sviluppo che supporta diversi sistemi operativi.
Per poter utilizzare il software di configurazione dovete montare il convertitore USB/Seriale, così da dialogare con la scheda tramite la seriale3 del microcontrollore, disponibile sul connettore UART3: è sufficiente cablare le linee TX,RX e GND; la linea +5V non è necessaria perché la scheda ha alimentazione propria. Il software è molto intuitivo e una volta avviato è sufficiente selezionare la seriale utilizzata per la comunicazione: il software provvederà, ad intervalli regolari, ad interrogare il robot. Tutti i principali dati interni sono visualizzati, mentre il valore dell'inclinazione, la corrente assorbita e la potenza dei motori sono anche visibili in un grafico. I campi in rosso permettono l'immissione dei parametri semplicemente trascinandovi sopra il puntatore del mouse. Il pulsante RESET consente il reset di tutti gli allarmi, OFFSET consente la determinazione dell'offset dei sensori, READ permette di leggere i parametri PID attualmente utilizzati, WRITE invia i parametri a video, mentre il pulsante SAVE consente di salvare nella memoria EEPROM del microcontrollore i parametri PID attualmente utilizzati.

Listato 5

```
void SetMotor()
{
  Steer = steerSensibilty * float(int(dataX)-127);
  Speed = speedSensibility * float(int(dataY)-127);

  drivePWM_L = int(  (drive*goStart) - Kdm*float(encoderDif) + Steer );
  drivePWM_R = int(  (drive*goStart) + Kdm*float(encoderDif) - Steer );

  // compensate motor dead band
  if (drivePWM_L>0) drivePWM_L += deadZoneMotor;
  if (drivePWM_L<0) drivePWM_L -= deadZoneMotor;
  if (drivePWM_R>0) drivePWM_R += deadZoneMotor;
  if (drivePWM_R<0) drivePWM_R -= deadZoneMotor;

  dutyCycleLeft = stallMotor + drivePWM_L;// + Steer;
  dutyCycleRight = stallMotor + drivePWM_R;// - Steer;

  dutyCycleLeft = constrain(dutyCycleLeft, stallMotor-maxPWM, stallMotor+maxPWM);
  dutyCycleRight = constrain(dutyCycleRight, stallMotor-maxPWM, stallMotor+maxPWM);
  OCR1B = dutyCycleLeft;   //set PWM SX 250+-250
  OCR1A = dutyCycleRight;  //set PWM DX 250+-250
}
```

CONFIGURAZIONE TELEMETRIA WIRELESS

La telemetria/configurazione wireless è decisamente più comoda, visto che il controllo completo del robot può avvenire in tempo reale mentre è in bilanciamento, è quindi semplice modificare i parametri al volo e vederne il comportamento in tempo reale.
La telemetria che consente la maggior portata può essere ottenuta con moduli XBee: anche la serie 2 da 2mW di potenza assicura una buona portata delle comunicazioni. I moduli XBee vanno programmati prima di essere utilizzati, perciò è necessario scaricare il software XCTU della Digi all'indirizzo *http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu*. Per la nostra applicazione è sufficiente che i due moduli comunichino in modalità *trasparent mode*. In questa modalità due moduli (e solo loro) possono sostituire a tutti gli effetti una comunicazione cablata. Per prima cosa inserite uno dei due moduli XBee nell'USB dongle

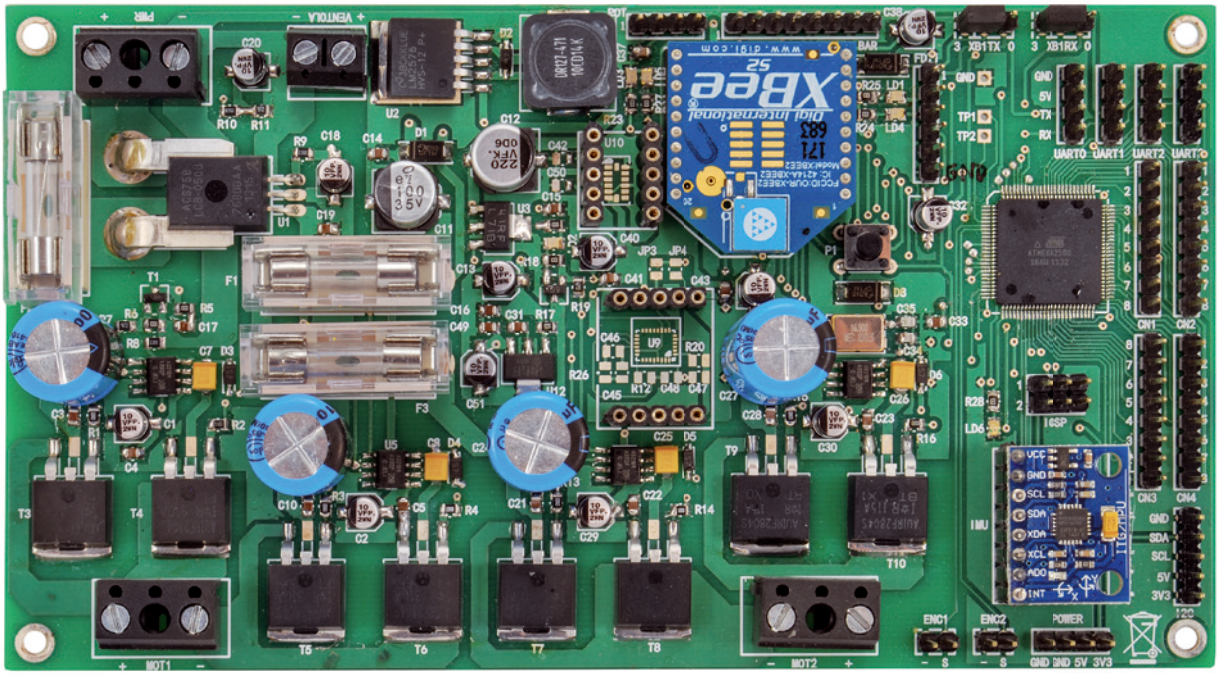
***Tabella 6** - Protocollo di scambio dati con OpenWheels2.*

Carattere ricevuto	Funzione svolta
A	risposta con tutti i parametri dei sensori e dello stato di funzionamento
E	risposta con i parametri PID attuali
W+(PID parameters)	ricezione parametri PID settati dall'utente
S	salvataggio dei parametri PID attuali in EEPROM
Z	set offset sensori
D+dataXY	ricezioni valori per comando movimentazione
C	reset errori ed encoder

***Tabella 7** - Segnalazioni di allarmi.*

Allarme	Stato
GO_Standby	avvisa di robot armato/standby
VBat_under	tensione di alimentazione troppo bassa
VBat_over	tensione di alimentazione troppo alta
IBat_over	corrente assorbita supera il limite impostato
Steer_error	errore lettura potenziometro dello sterzo (solo per cloni di SegWay)
Acc_error	errore lettura accelerometro
Gyro_error	errore lettura giroscopio
Angle_error	la misura dell'angolo eccede i limiti ammessi
EEPROM_error	si è verificato un errore nella lettura dei dati in EEPROM*
Speed_error	la velocità delle ruote eccede il valore massimo

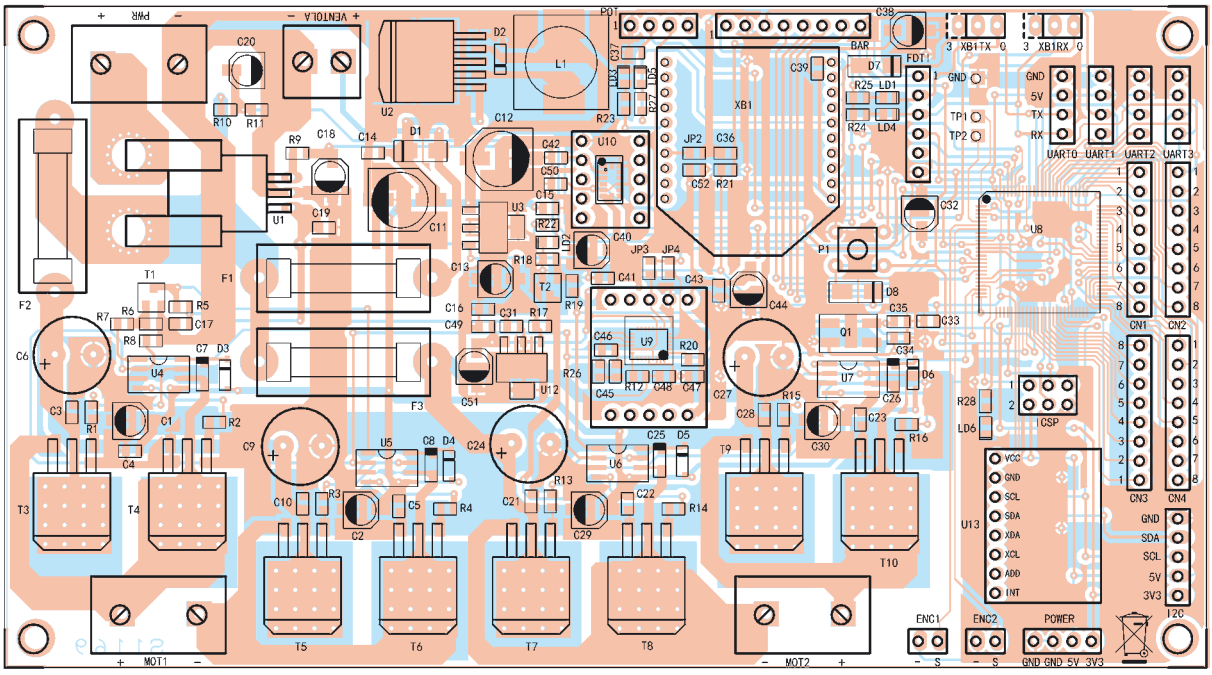
** alla prima messa in servizio non vi sono dati in memoria quindi si verifica l'errore.*



Elenco Componenti:

R1÷R4, R7: 4,7 ohm (0805)
R5, R6: 1 kohm (0805)
R8, R9: 2,2 kohm (0805)
R10: 47 kohm (0805)
R11, R13÷R16, R19: 4,7 ohm (0805)
R12: 33 kohm (0805)
R17, R18: 1 kohm (0805)
R20: 33 kohm (0805)
R21, R26: 10 kohm (0805)
R22÷R25: 1 kohm (0805)
R27, R28: 1 kohm (0805)
C1, C2: 10 µF 35 VL elettrolitico (B)
C3÷C5, C10: 100 nF ceramico (0805)
C6, C9: 1000 µF 35 VL elettrolitico
C7, C8: 10 µF 25 VL tantalio (A)
C11: 100 µF 35 VL elettrolitico (F)
C12: 220 µF 35 VL elettrolitico (F)
C13: 10 µF 35 VL elettrolitico (B)
C14÷C17: 100 nF ceramico (0805)
C18: 10 µF 35 VL elettrolitico (B)
C19: 100 nF ceramico (0805)
C20: 10 µF 35 VL elettrolitico (B)
C21÷C23: 100 nF ceramico (0805)
C24, C27: 1000 µF 35 VL elettrolitico
C25, C26: 10 µF 25 VL tantalio (A)
C28, C31: 100 nF ceramico (0805)
C29, C30: 10 µF 35 VL elettrolitico (B)
C32: 10 µF 35 VL elettrolitico (B)
C33: 100 nF ceramico (0805)
C34, C35: 10 pF ceramico (0805)
C36, C39, C41, C43: 100 nF ceramico (0805)
C37: 220 nF ceramico (0805)
C38, C40: 10 µF 35 VL elettrolitico (B)
C40: 10 µF 35 VL elettrolitico (B)
C42: 3,3 nF ceramico (0805)
C44: 10 µF 35 VL elettrolitico (B)
C45: 470 nF ceramico (0805)
C46: 10 nF ceramico (0805)
C47÷C49: 100 nF ceramico (0805)
C50: 3,3 nF ceramico (0805)
C51: 10 µF 35 VL elettrolitico (B)
C52: 100 nF ceramico (0805)

ed impostate i parametri ID=0 e MY=0, quindi leggete i valori SH e SL. A questo punto scambiate i due moduli, trascrivete nuovamente i valori SH e SL ed impostate sui campi DH e DL i valori letti sul precedente modulo: DH=SH e DL=SL. A questo punto scambiate nuovamente i due moduli ed inserite nel campo DH e DL del primo modulo i valori SH e SL del secondo modulo. In questo modo l'indirizzo di destinazione (D) del primo modulo coinciderà con l'indirizzo fisico del secondo modulo (S) e viceversa. A questo punto i due moduli sono pronti a comunicare. Non dimenticate di posizionare i due jumper XB1TX e XB1RX verso il "3" per mettere in comunicazione il modulo con la Seriale3 dell'Atmel adibita alla telemetria. Installate il secondo modulo su Openwheels2 e lasciate il primo connesso al PC, avviate il software OpenWheelsGUI_V20 e selezionate la porta seriale relativa al dongle XBee: partirà immediatamente la lettura dei parametri. È possibile visualizzare tutti i parametri di funzionamento compreso un grafico in tempo reale dell'angolo di inclinazione, della potenza sui motori e della corrente assorbita. Una riga di testo in basso riporta lo stato di funzionamento ed eventuali messaggi di errore.
Esiste anche la possibilità di utilizzare la telemetria via Bluetooth: in questo caso occorre attrezzarci di conseguenza dotando la scheda Openwheels 2 con un modulo BT. Per evitare di dover mettere mano allo sketch abbiamo utilizzato

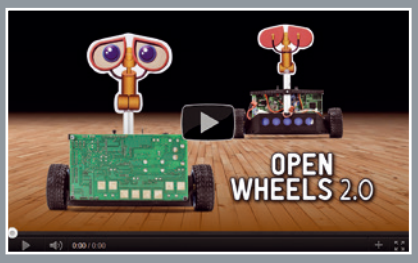


D1: GF1M
D2÷D6: BAT42W
D7, D8: GF1M
D8: GF1M
T1: BC817
T2: BC817
T3÷T10: AUIRF2804S
Q1: Quarzo 16 MHz
U1: ACS758LCB-050U-PFF
U2: LM2576HVS-12/NOPB
U3: LM2937IMP-5.0/NOPB
U4÷U7: IRS2184S
U8: ATMEGA2560
U9: LPY503AL
U10: MMA7361
U11: XBEE1MWCHIP
U12: TC1262-3.3VDB
U13: MPU6050 (GY-521)
L1: Induttanza 470 μH 850 mA
F1: Fusibile 1A
F2, F3: Fusibile 10A
LD1, LD2: LED verde (0805)
LD3÷LD5: LED rosso (0805)
LD6: LED giallo (0805)
P1: Microswitch

Varie:
- Porta fusibile da CS (3 pz.)
- Morsettiera 2 poli passo 5 mm
- Morsettiera 2 poli passo 10 mm (3 pz.)
- Strip maschio 2 poli (2 pz.)
- Strip maschio 3 poli (4 pz.)
- Strip maschio 4 poli (6 pz.)
- Strip maschio 5 poli (1 pz.)
- Strip maschio 6 poli (1 pz.)
- Strip maschio 8 poli (5 pz.)
- Strip femmina 5 poli (4 pz.)
- Strip femmina 8 poli (1 pz.)
- Strip femmina 10 poli passo 2 mm (2 pz.)
- Jumper (2 pz.)
- Circuito stampato S1169

un modulo Bluetooth che una volta configurato non richiedesse ulteriori istruzioni per il funzionamento, come il modulo DG010 della Dagurobot, che va connesso sempre alla Seriale3 tramite il connettore UART3 (**Fig. 10**). Questo modulo viene venduto preconfigurato per il funzionamento con protocollo SPP (Serial Port Profile) alla velocità di 9.600 bps; l'SPP permette al modulo di trasferire serialmente dei dati esattamente come farebbe un cavo e analogamente al funzionamento dei moduli XBee in transparent mode. Per la nostra applicazione è necessario impostare una velocità di comunicazione a 115 kbps e il nome della periferica in "PRobot"; allo scopo bisogna interfacciare il modulo al PC tramite il convertitore USB/Seriale. Usate un software che possa inviare stringhe tramite la seriale come CoolTermWin oppure lo stesso SerialMonitor di Arduino, impostando la velocità di comunicazione a 9.600 bps. Inviate quindi la stringa AT senza aggiungere alcun fine riga o ritorno a capo: se il modulo risponde con OK significa che la comunicazione è stabilita. Modificate il nome del dispositivo con il comando AT+NAMEPRobot e la velocità di comunicazione con il comando AT+BAUD8. A questo punto il modulo è configurato e pronto a funzionare nella modalità richiesta. Anche il PC dovrà disporre della comunicazione Bluetooth; applicando il dongle è possibile abilitare alla comunicazione Bluetooh qualsiasi PC. Se riscontrate problemi con i driver

Guarda OpenWheels in azione su YouTube!



Vuoi vedere il nostro balancing robot in azione? Nulla di più semplice: collegati alla pagina *https://youtu.be/FagvW4DdBPI*. Dal nostro canale YouTube *https://www.youtube.com/user/ElettronicaIN/videos* potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato come il nostro segway opensource OpenWheels.

di Windows, potete installare il software Bluesoleil della IVT Corporation. Non appena alimenterete il modulo il LED rosso si accenderà a luce fissa, mentre il LED blu lampeggerà; non appena vi sarete connessi al modulo il LED blu rimarrà acceso a luce fissa, ad indicare l'abilitazione alla trasmissione. Per utilizzare la telemetria per PC è sufficiente specificare nel software la porta seriale assegnata alla comunicazione SPP del dongle BT.

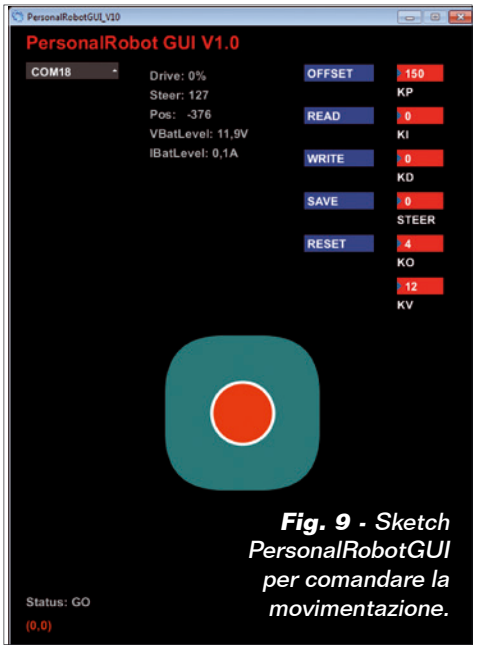


***Fig. 9** - Sketch PersonalRobotGUI per comandare la movimentazione.*

MESSA IN SERVIZIO

Con il software di telemetria avviato, è opportuno impostare tutti i parametri del controllo PID a zero e procedere alla determinazione dell'offset; posizionate il robot esattamente in equilibrio in modo stabile e senza vibrazioni e premete il pulsante OFFSET: la taratura è automatica e richiede due secondi. Per evitarvi di ripetere l'operazione ad ogni accensione, i parametri vengono salvati nella memoria permanente del micro. Ora portate il robot dallo stato di stand-by a quello di Go e resettate eventuali errori; non vi resta che agire sui parametri del PID per ottenere la stabilizzazione. Non esiste una procedura certa e sicura per determinare immediatamente i parametri corretti; il più delle volte serve molta pazienza e diverse prove per arrivare ad una stabilizzazione soddisfacente. La procedura ideale prevede che, partendo con tutti i parametri nulli, si alzi il solo parametro KP finché il robot non cominci ad oscillare; successivamente si abbassa il valore di KP in modo che il robot non oscilli, quindi si alza il valore di KV. Già questi due parametri dovrebbero garantire la stabilizzazione e solo in un secondo momento si alza il valore di KO per assicurare che il robot mantenga la posizione. In un secondo momento potete provare ad alzare anche il valore di KD per verificare se la stabilizzazione migliora.
Con il robot da noi proposto è stato sufficiente impostare: KP=160, KO=5 e KV=12 (i rimanenti parametri sono a zero).

CONTROLLO REMOTO

Non poteva mancare la possibilità di controllare a distanza la movimentazione del robot: la filosofia di base è ingannare il robot facendogli credere di essere perfettamente bilanciato, anche se in realtà è leggermente (parliamo di decimi di grado) inclinato, pertanto tendente a cadere.
È sufficiente aggiungere o togliere un determinato valore alla variabile Pvel affinché il robot avanzi o indietreggi, mentre, per farlo sterzare, dobbiamo modificare il valore del PWM dei due motori incrementando l'uno e diminuendo l'altro, in modo che la somma rimanga costante. Sommando i due effetti si può far virare il robot e quindi comandarlo in

per il MATERIALE

I prodotti presentati in questo progetto possono essere acquistati presso Futura Elettronica. La board Open Wheels 2.0 (cod. FT1169M) costa 145,00 Euro, la batteria Lipo 1800 mAh/11,1V (cod. LIPO1800) è disponibile a 29,00 Euro. Il modulo Xbee 1 mW con antenna integrata (cod. XBEE1MWTANT) costa 36,00 Euro, quello da 2 mW completo di antenna (cod. XBEE2MW) 36,00 Euro, mentre l'Explorer Dongle V2 (cod. XBEEDONGLE2) è in vendita a 33,50 Euro. Il modulo Bluetooth controller DAGU (cod. DG010) è disponibile a 35,00 Euro, il convertitore FTDI USB-Seriale 3,3V/5V (cod. FTDI5V) costa 19,00 Euro, il motoriduttore con encoder e ruote (cod. BALANCINGBOT) è in vendita a 74,00 Euro.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 · Fax: 0331-792287
http://www.futurashop.it

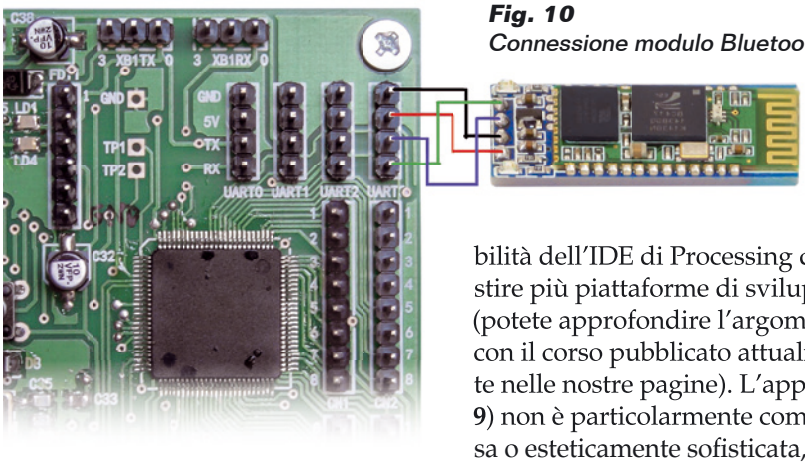


Fig. 10
Connessione modulo Bluetooth.

qualsiasi direzione con qualsiasi velocità (nei limiti consentiti dalla meccanica e della fisica). Come già accennato i parametri KO e Kdm devono essere posti a zero altrimenti il robot interpreterà i comandi di movimento come dei disturbi e cercherà di compensarli. Anche in questo caso abbiamo scritto un apposito software di nome PersonalRobotGUI_V10 che permette di mantenere sotto controllo i principali parametri e nel contempo di comandare il movimento tramite un semplice joystick virtuale, da azionare con il mouse.
Vista la criticità del robot, i comandi dovranno essere graduali e fluidi e di certo non ci si può aspettare di avere un controllo immediato come quello di un robot a quattro ruote.

CONTROLLO DA ANDROID

Abbiamo previsto la possibilità di comandare il robot tramite un'app Android, scritta grazie alla possibilità dell'IDE di Processing di gestire più piattaforme di sviluppo (potete approfondire l'argomento con il corso pubblicato attualmente nelle nostre pagine). L'app (**Fig. 9**) non è particolarmente complessa o esteticamente sofisticata, ma fa il suo dovere, ricevendo i dati di telemetria e inviando i dati del joystick virtuale, implementato sul display e facilmente azionabile con il pollice. L'App è stata scritta per uno smartphone Samsung S4 con display FHD e display AMOLED, per cui abbiamo preferito uno sfondo di colore molto scuro per evitare un eccessivo consumo di corrente durante il funzionamento. Per utilizzare lo smartphone, come telecomando dovete installare l'App di nome *PRobot.apk* che troverete tra i file del progetto. Non essendo stata prelevata dallo store ufficiale, l'installazione potrà avvenire solo se è stata attivata la modalità di installazione da sorgenti sconosciute, di solito disponibile nella sezione sicurezza delle impostazioni.
Per prima cosa dovete associare la periferica Bluetooth accedendo alla sezione BT nelle impostazioni del telefono (la password è 1234); a questo punto potete avviare l'App, che è configurata per ricercare la periferica PRobot tra quelle associate e collegarvi a essa (**Fig. 14**). Non serve connettersi ad alcun altro dispositivo. Instaurata la connessione, sul display saranno visualizzati i dati di telemetria e potrete azionare il joystick per comandare il robot.
Bene, con questo abbiamo detto tutto; ora sta a voi dare forma al progetto, variando la meccanica o il firmware a piacimento.

Fig. 11
Configurazione modulo Bluetooth.

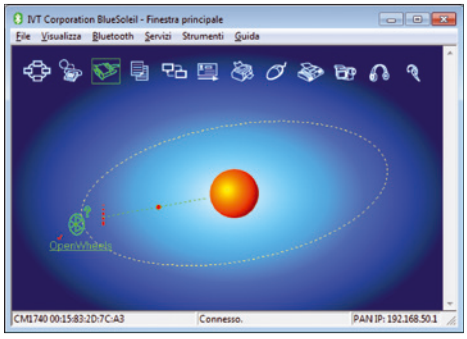
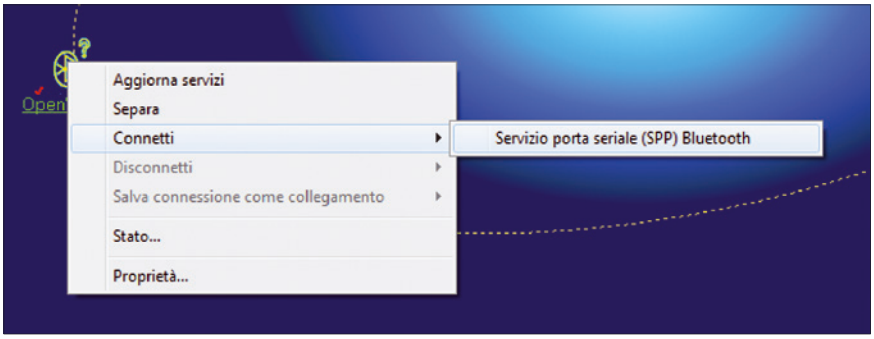


***Fig. 12** - Software BlueSoleil per il Bluetooth.*

***Fig. 13** - Connessione al modulo Bluetooth da PC.*



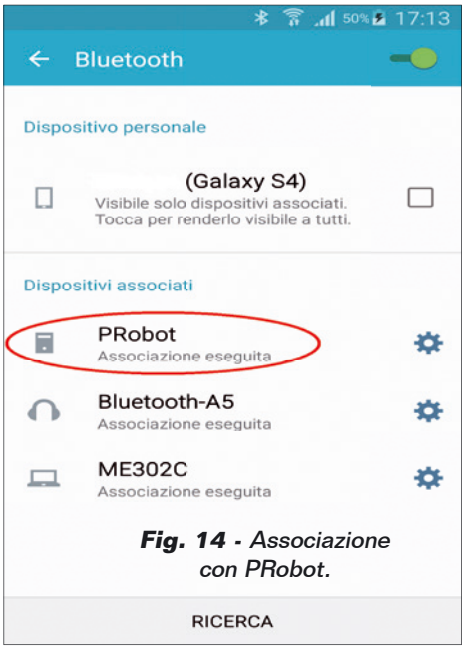


***Fig. 14** - Associazione con PRobot.*