KIPR Open Robot 2 Documentation

Table of Contents

**KIPR Board**3

Schematic3

Layout4

**Control Systems**5

Motor Controller5

PSoC8

**Raspberry Pi** 9

Board 9

Code Overview 9

**Robot Overview** 10

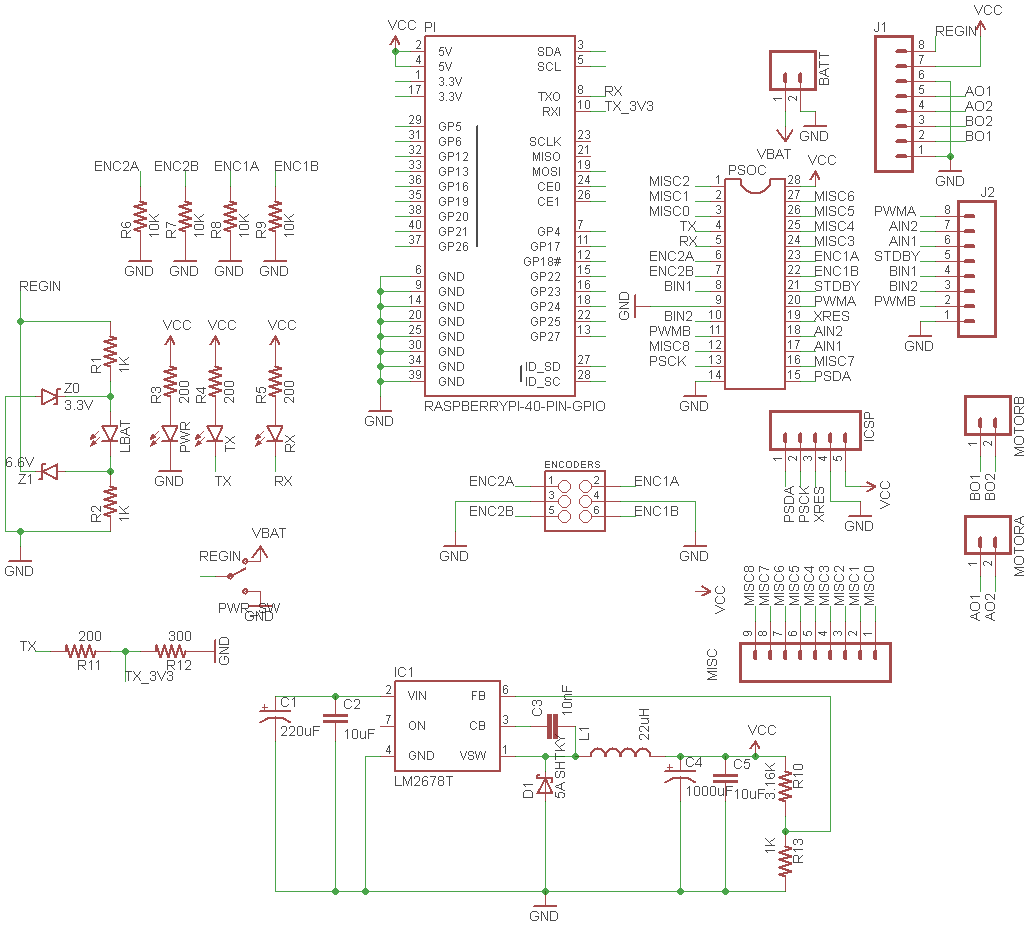
Mechanical 11

Electrical 12

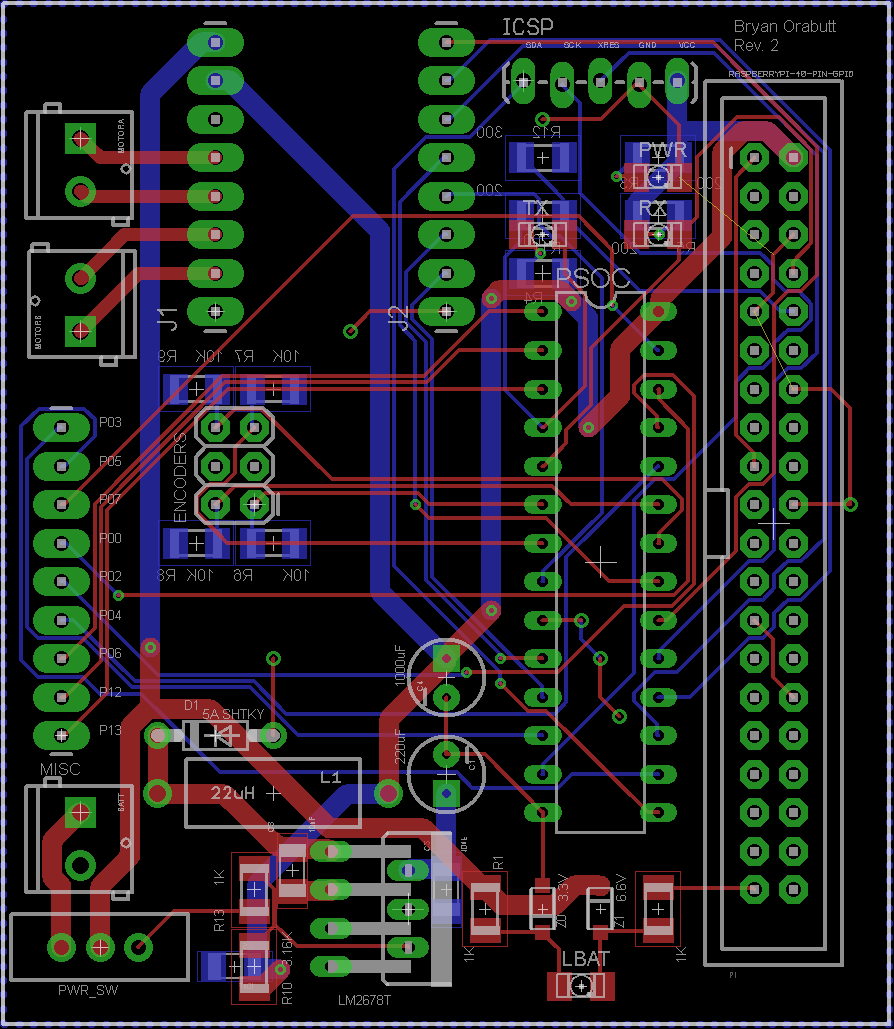
Functionality 13

# **KIPR Board**

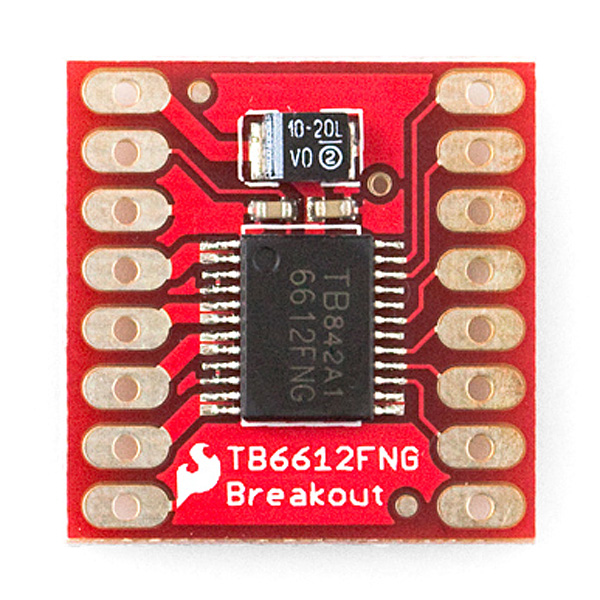
# **PCB Schematic**



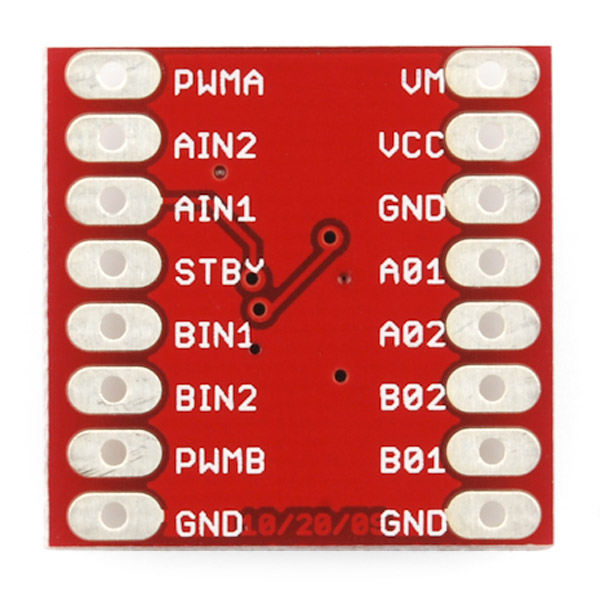
**Board Layout**



**Control Systems**

**Motor Controller**

Top side

Bottom Side****

The TB6612FNG motor driver can control up to two DC motors at a constant current of 1.2A (3.2A peak). Two input signals (IN1 and IN2) can be used to control the motor in one of four function modes - CW, CCW, short-brake, and stop. The two motor outputs (A and B) can be separately controlled, the speed of each motor is controlled via a PWM input signal with a frequency up to 100kHz. The STBY pin should be pulled high to take the motor out of standby mode.

Logic supply voltage (VCC) can be in the range of 2.7-5.5VDC, while the motor supply (VM) is limited to a maximum voltage of 15VDC. The output current is rated up to 1.2A per channel (or up to 3.2A for a short, single pulse).

The board can be ordered form Sparkfun Electronics, product SKU 09457.

The board pinout is as follows:

**VCC:** 5V

**GND:** Common GND

**VM:** Motor supply voltage

**PWMA/B:** Pulse width modulated signal for controlling speed.

**A01/A02:** Motor A connections

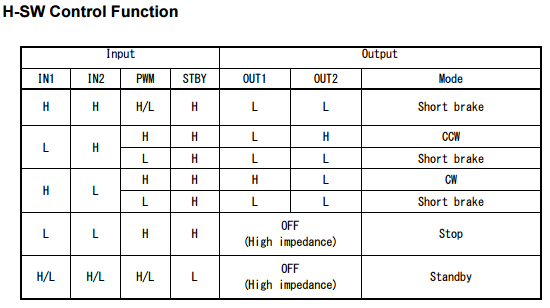
**B01/B02:** Motor B connections

**AIN1/AIN2:** Motor A control pins

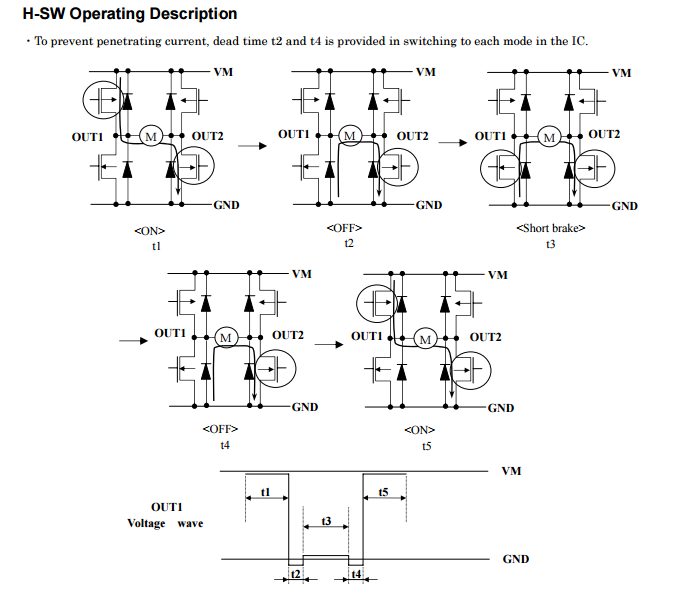
**BIN1/BIN2:** Motor B control pins

**STBY:** Stops motors and places controller in low power state.

Control States:

****

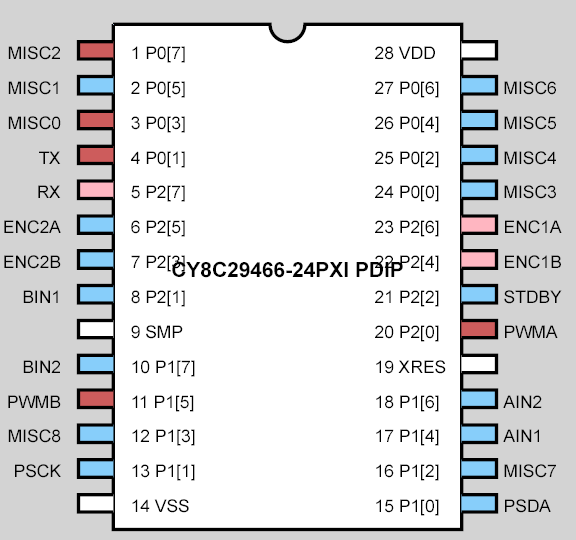
HW Diagram

****

**PSoC**

The PSoC CY8C29466-24PXI is the brain of the control systems. It reads the encoders, communicates with the Motor Controller, and handles serial messages with the Raspberry Pi. There is a range of functions that the PSoC has been programmed to perform and they can all be accessed via serial messages from the Raspberry Pi.

PSoC Pinout:



Functional Description of Pins:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Functional Description** | **Pin Name** | **Functional Description** |
| Misc 0 | Allocated to Servo1 PWM module | VDD | 5V Supply |
| Misc 1 | Unused, available for use upon reprogramming | VSS | GND |
| Misc 2 | Allocated to Servo0 PWM module | ENC1A | A output of encoder1 |
| Misc 3 | Unused, available for use upon reprogramming | ENC1B | B output of encoder1 |
| Misc 4 | Unused, available for use upon reprogramming | ENC2A | A output of encoder2 |
| Misc 5 | Light Sensor input | ENC2B | B output of encoder2 |
| Misc 6 | Unused, available for use upon reprogramming | STDBY | Motor control STDBY, unused |
| Misc 7 | Debug pin, also available for reprogramming\* | PWMA | Motor control PWMA, allocated to a PWM module |
| Misc 8 | Debug pin, also available for reprogramming\* | PWMB | Motor control PWMB, allocated to a PWM module |
| RX | Serial receive line, goes to Raspberry Pi TX | AIN1 | Motor control AIN1, allocated to output pin |
| TX | Serial transmit line, goes to Raspberry PI RX through 2/3 divider | AIN2 | Motor control AIN2, allocated to output pin |
| PSCK | I2C clock, used for programming | BIN1 | Motor control BIN1, allocated to output pin |
| PSDA | I2C data, used for programming | BIN2 | Motor control BIN2, allocated to output pin |
| XRES | External Reset, used for programming | SMP | PSoC switch mode pump, unused |

\*Debug pins can be repurposed without disrupting the debug functionality.

**Debug Mode**

Debug mode will allow the board to operate as normal, however every serial command sent will echo back useful debugging data. To enable debug mode connect each debug pin to +5V before powering the board.

Available serial commands:

|  |  |  |
| --- | --- | --- |
| **Command** | **Packet** | **Description** |
| MAV | "a**<p>**\a" | Set robot velocity to **<p>** RPS |
| MOV | "b**<p>**\a" | Set motor PWMs to pulse width **<p>** (8-bit) |
| GETV | "c\a" | Gets the current velocity of the robot in RPS |
| SERV0\_POS | "d**<p>**\a" | Sets servo0 PWM to pulse width **<p>** (16-bit) |
| SERV1\_POS | "e**<p>**\a" | Sets servo1 PWM to pulse width **<p>** (16-bit) |
| SRV0\_STP | "f\a" | Stops servo0 PWM |
| SRV1\_STP | "g\a" | Stops servo1 PWM |
| GETC1 | "h\a" | Get's the cumulative running count of encoder 1 |
| GETC2 | "i\a" | Get's the cumulative running count of encoder 2 |
| RSTC1 | "j\a" | Resets running count of encoder 1 to 0 |
| RSTC2 | "k\a" | Resets running count of encoder 2 to 0 |
| STOP | "l\a" | Stops the robot |
| FORWARD | "m\a" | Sets motor directions to give forward movement |
| BACKWARD | "n\a" | Sets motor directions to give backward movement |
| TURNL | "o\a" | Sets motor directions to give left rotation. |
| TURNR | "p\a" | Sets motor directions to give right rotation. |

Notes: all packets are string types.

\a denotes ASCII BEL character 0x07

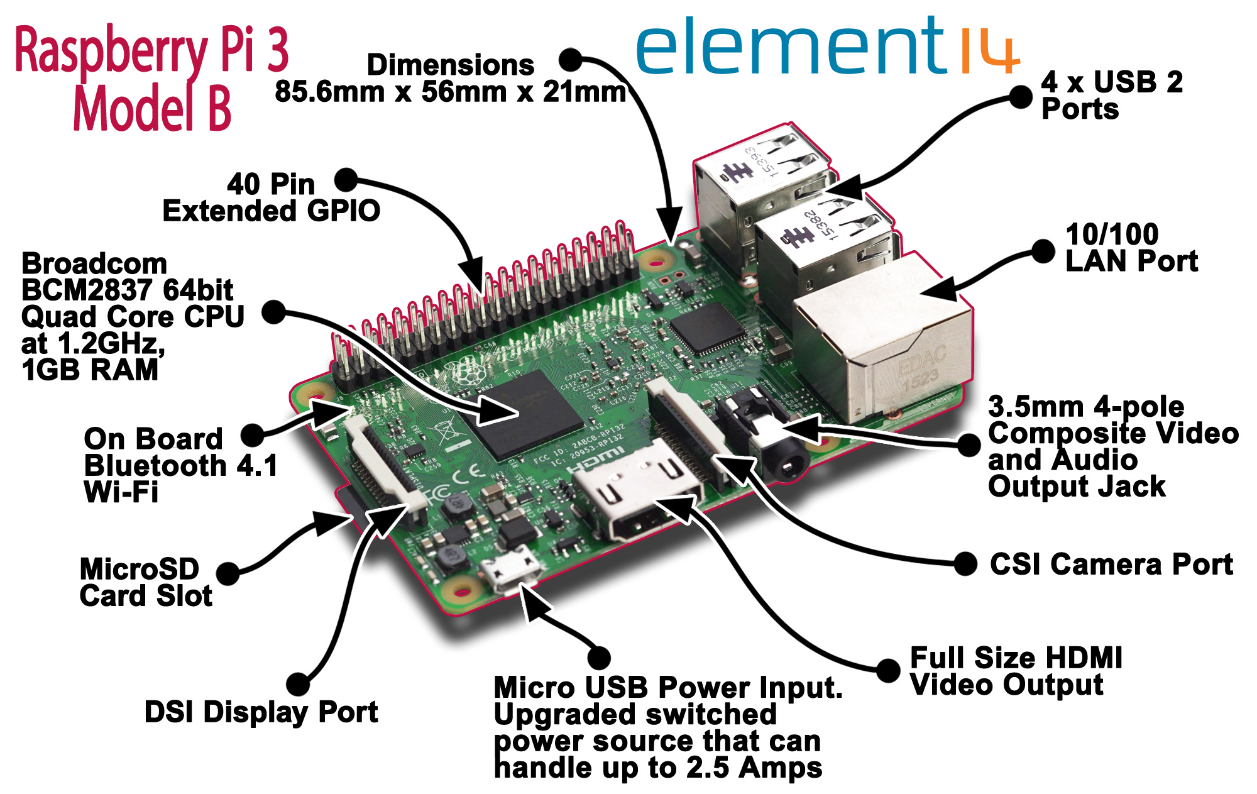
**<p>** is a command parameter, replace with an appropriate value.

16-bit servo pulse widths must be in range 750-2400 (hardware reasons)

GETV uses costly floating point calculations. Do not use more often than needed.

**Raspberry Pi 3**

**Board**

****