Gibbot v1 Wixel

Matthew Collins

July 22, 2014

Contents

1 Wixel **3**
1.1 Wixel Basics 3
1.2 Programming the Wixel 4

2 dsPIC **4**

3 Encoders **5**
3.1 Motor Encoder 5
3.2 Wall Encoder 5

4 H-bridge/Current Sensor **5**

5 Motor and Bevel Gear **6**

6 Future Work **7**

1 Wixel

1.1 Wixel Basics

The Wixel is a programmable microcontroller created by Pololu. The central component of the Wixel is a CC2511F32 microcontroller from Texas Instrument. For additional functionality, a 2.4 GHz radio transceiver and USB interface were added. It has 15 I/O lines, which features 6 analog inputs, 2 USARTS, and 7 timer channels (capable of complimentary mode PWM)[1]. The Wixel essentially compresses the necessary capabilities of two chips (i.e. a dsPIC and XBee in the case of the Gibbot PCB) into just one.

The schematic for the Wixel, as well as the breakout board pinout, can be seen in Figure 1. Note: for the pinout, black text indicates default pin assignments while white text indicates secondary location.

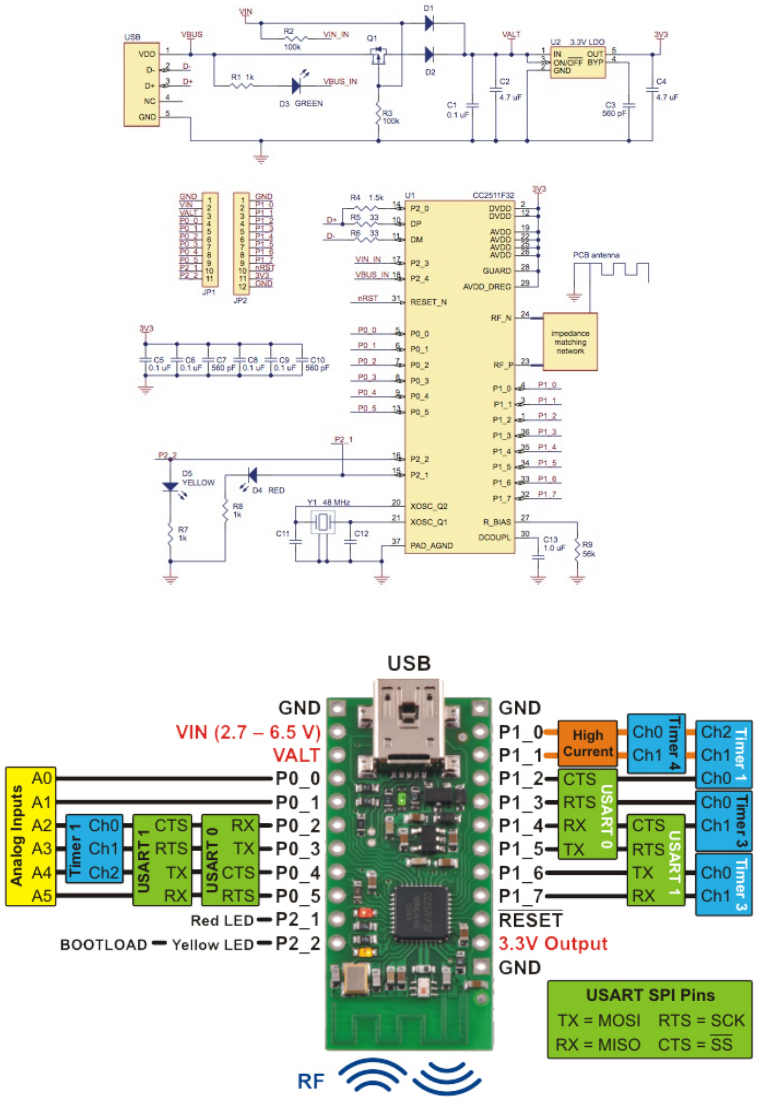

Figure 1: (Top) Wixel Schematic (Bottom) Wixel Pinout

[1] Pololu Wixel Page http://www.pololu.com/product/1336

1.2 Programming the Wixel

Configuring Pins and Peripherals

To configure the pin functions on the Wixel, the correct bits on the TI CC2511F32 must be programmed. These include configuring the pins to inputs/outputs, ADC inputs, and setting timers and the corresponding PWM pins and their mode (complementary, etc.). All the available chip functions and the configuration bits can be found in the TI CC2511F32 datasheet.

Wixel Peripheral Functions

After configuring the pins, code can be written that utilizes the specified functions. Some of the additional components on the Wixel not contained within the TI chip, including the radio antenna and USB interface, cannot be directly programmed through the TI datasheet. There are a some premade libraries of functions, however, that allow for the control of all facets of the Wixel. These include: a radio library, USB library, a peripheral library, and a basic Wixel library. These libraries and the functions they contain can be found on the Wixel SDK.

Notes on Programming the Wixel

- To use many of the libraries, the corresponding .lib file must be included in the options.mk document in the program folder.
- Changing the location of functions to their secondary location must be done through the configuration bits. The locations (primary and secondary) of all the Wixel's functions can be seen in Figure 1.
- The timeInit() function in the time.c file (wixel.lib) uses the timer 4 ISR. If the timer 4 ISR is needed, do not call the timeInit() function. The systemInit() function calls the timerInit() function, so if the T4 ISR is needed, this function should be avoided. The initialization functions can be found on the SDK under board.h within the basic Wixel library.

2 dsPIC

The dsPIC is used to get data from the two encoders. Two 16-bit QEI readers are used on the dsPIC, one for the motor encoder and another for the wall encoder. The QEI is set to start at the center of its count (i.e. at 32,766). Moreover, the dsPIC is configured to count all four pulses from the encoders (up and down from channel A and the same for channel B). Since the encoder readings from the dsPIC are needed, it must be able to communicate with the Wixel. This is done through UART. The pinout for the dsPIC can be seen in Table 1.

	Channel A/TX Pin	**Channel B/RX Pin**
Motor Encoder	B7	B8
Wall Encoder	B1	B2
UART	B14	B12

Table 1: dsPIC Pinout

For additional information about how the dsPIC is being used, read the dsPIC code.

3 Encoders

3.1 Motor Encoder

The motor has an attached optical encoder to provide feeback on the motor's position. There are 500 counts per rotation; however, there is a 19.5:1 internal gear ratio that needs to be considered. Moreover, the bevel gear (see Section 5) that connects the motor to the second link has a gear ratio of 3:1. This gives a dsPIC count of:

$$CPR = 500 * 19.5 * 4 * 3 = 117,000$$

Where the 500 is the CPR of the motor, the 19.5 is from the gear ratio, and the 4 is for the way the dsPIC is counting (see Section 2 for more on dsPIC) and the 3 is the bevel gear ratio.

3.2 Wall Encoder

The encoder used at the wall connection is a CUI AMT 10 Modular Encoder. It is a capacitive encoder and allows for 16 different resolution settings depending upon the postion of four dip switches.[2] The possible resolution settings, as well as a picture of the dip switches, can be seen in Figure 2.

1 = On, 0 = Off

Resolution (PPR)	Maximum RPM	1	2	3	4
2048	7500	0	0	0	0
1024	7500	0	0	1	0
1000	7500	1	0	0	0
800	7500	0	1	0	0
512	15000	0	0	0	1
500	7500	1	0	1	0
400	7500	0	1	1	0
384	7500	1	1	0	0
256	15000	0	0	1	1
250	15000	1	0	0	1
200	15000	0	1	0	1
192	7500	1	1	1	0
125	15000	1	0	1	1
100	15000	0	1	1	1
96	15000	1	1	0	1
48	15000	1	1	1	1

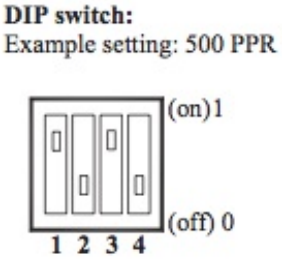

Figure 2: Wall Encoder Resolution Settings

The current resolution being used is the maximum of 2048 PPR to most accurate track the position. To change the resolution, open the plastic cover on the encoder and move the dip switches to the desired postions.

4 H-bridge/Current Sensor

Since the voltage supply needed for the motor exceeds the 5V used for the other applications, a secondary power source and driver is needed to run the motor. This is done through the MC33926 Motor Driver Carrier on a breakout board by Pololu. This supplies one full H-bridge to drive the motor between the two members that can handle the necessary voltage and current requirements. The H-bridge has an operating range of 5-28V, but for this project, since a 24V motor is being used, the input voltage is 24V. Moreover, it can support up to 20kHz PWM and peak current of up to 5A for a few seconds.[3]

[2] http://www.cui.com/product/resource/amt10.pdf
[3] http://www.pololu.com/product/1212

In addition to driving the motor, the MC33926 Motor Driver Carrier also contains a feedback pin. This acts as a high-side unidirectional current sensor that provides analog current-sense feedback. The Pololu breakout board has been setup to provide feedback of approximately 525mV per amp.

The pinout for the Pololu breakout board of the MC33926 Motor Driver Carrier can be seen in Figure 3. Note: the default-overriding jumpers are not being used.

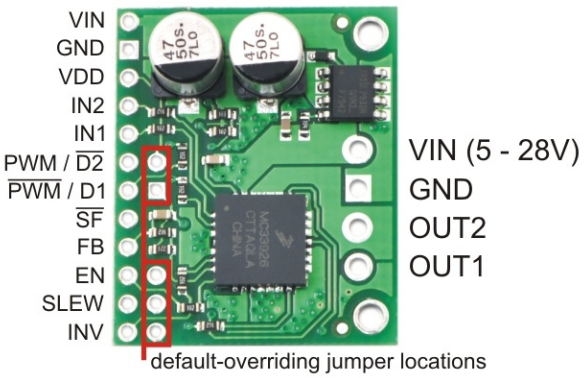

Figure 3: H-bridge Pinout

The INV pin inverts the meaning of the two PWM inputs to the driver when logical high. This allows the INV pin to function as a direction line. This pin is being used to switch the direction that the motor spins.

For more information and the input conditions for different outputs (forward, reverse, free wheeling, etc.) can be found in the MC33926 datasheet.

5 Motor and Bevel Gear

The motor used is the Pittman GM8724 motor with a 19.5:1 gear ratio and 91X0 encoder (See Series GM 8000 LO-COG Brush Commutated DC Motors section in datasheet). The motor is used to drive a bevel gear (see Figure 4 for setup). The gears are setup at a 90 degree angle to change the direction of the shaft's rotation, thus allowing the motor to sit in the plane of the link and to reduce the necessary thickness of the link.

Note: The bevel gear assembly has a 1:3 gear ratio that needs to be taken into account for the motor encoder.

Figure 4: Motor and Bevel Gear in Frame

6 Future Work

IR LEDs

The six Optitrack IR LEDs used for tracking the Gibbot using cameras still have to be wired up to the board and connected to the frame (holes already cut into frame for LEDs).

Swing-up and Controls

A swing-up algorithm needs to be written to control the gibbot. After perfecting this, coding for more advanced movement sequences has to be written.