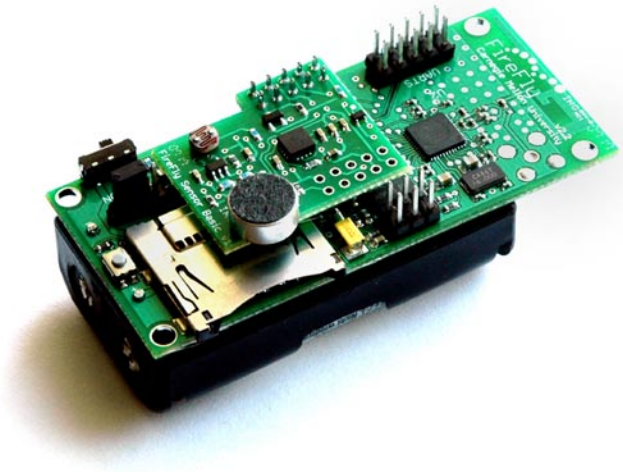


FireFly 2.2 Datasheet

August 20, 2007

This page intentionally left blank.

Contents

1. INTRODUCTION 1
Features 1
Applications 2
2. BLOCK DIAGRAM 3
3. HARDWARE CONNECTIONS 4
Power 5
Header 1 (UARTS, I2C, GPIO, ISP) 6
Header 2 (ADC) 7
Header 3 (SPI / ISP) 8
Header 4 (GPIO) 9
Antenna Configurations 10
LEDs 11
Button 12
4. HARDWARE CHARACTERISTICS 13
5. SENSORS 14
6. USING THE FIREFLY PROGRAMMER 15
Connecting the Programmer 15
Compiling Source 18
Setting FireFly Fuses (first time only) 18
Downloading the Code 19
7. ASSEMBLY INFORMATION 20
Schematic 20
Dimensions 24

This is the FireFly 2.2 Datasheet v0.90 for the FireFly Wireless Sensor Networking Platform.
For more information go to http://www.nano-rk.org/FireFly
Copyright 2007 Anthony Rowe and the Real-Time And Multimedia Lab at Carnegie Mellon University. All Rights Reserved.

1. INTRODUCTION

The FireFly Sensor Networking Platform is a low-cost low-power hardware platform for wireless sensor networking. In order to better support real-time applications, the system is built around maintaining global time synchronization. The main Firefly board uses an Atmel ATmega1281 8-bit micro-controller with 8KB of RAM and 128KB of ROM along with Chipcon's CC2420 IEEE 802.15.4 standard-compliant radio transceiver for communication. The maximum packet size supported by 802.15.4 is 128 bytes and the maximum raw data rate is 250Kbps. Most FireFly boards come with a sensor expansion card that provides light, temperature, audio, three axis acceleration and voltage sensing. The base FireFly platform provides an SDIO port which can be used for large FLASH storage or as a universal interface to PC compatible peripherals. The FireFly platform includes a downloading and code execution time profiling (TimeScope) environment for easy development.

Features

- 802.15.4 Zibee Compatible Radio
- Low Power ATmega1281 processor
  - 8 KB Ram, 128 KB Rom, 2 KB EEPROM
  - 7.3728 MHz
  - 2 UARTS, 1 I2C, 1 SPI
- Mini-SD card slot
  - FLASH storage
  - Standard Peripheral Expansion Slot
- Sensor Expansion Card
  - Light
  - Temperature
  - 3-Axis Acceleration
  - Audio
  - Battery Level Indicator
- Nano-RK RTOS
  - C GNU tool-chain
  - Classical Preemptive Operating System Multitasking Abstractions
  - Real-Time Priority Based Scheduling
  - Built-in Fault Handling
    - Task Timing Violations
    - Stack Integrity
    - Unexpected Node Restarts
    - Resource Over-Use
    - Low Voltage Detection

  Watchdog Timer
- PCB or External Antenna
- Selectable Voltage Regulator
- User Defined Push Button

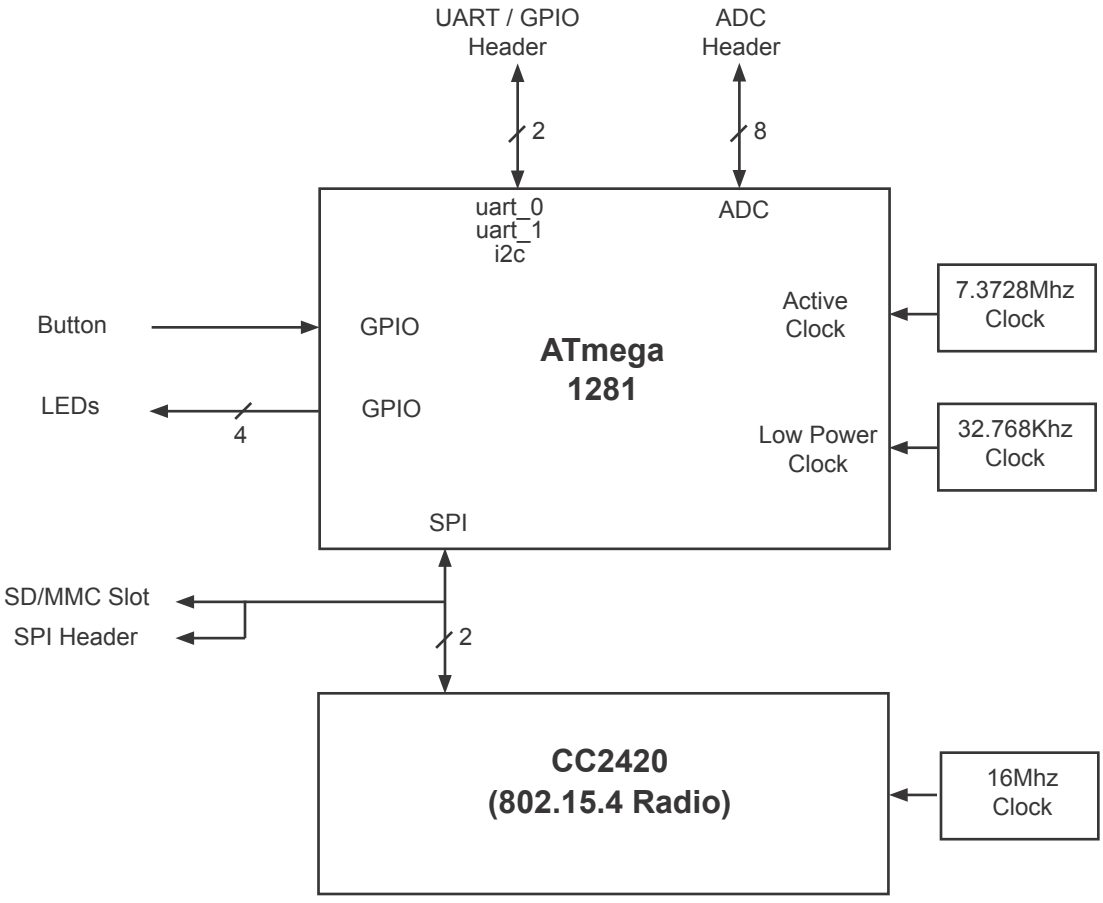
Applications

The FireFly sensor networking platform is designed for real-time sensor networking applications including:

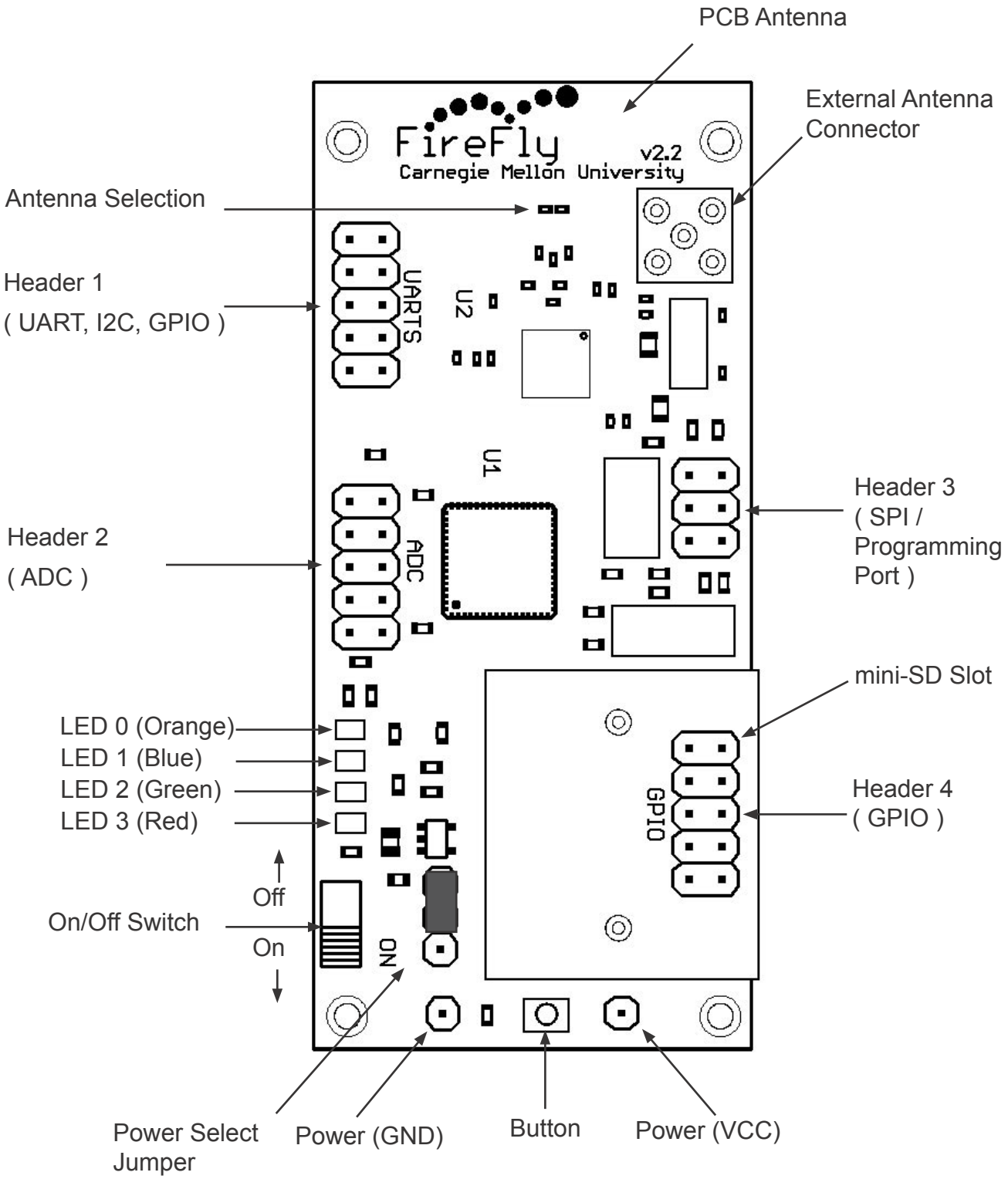
- Industrial Control and Automation
- Smart Home Monitoring
- Inventory and Personnel Tracking Systems
- Embedded System Education
- Hazardous Environment Monitoring

2. BLOCK DIAGRAM

Below is a high level block diagram of the major components found on the FireFly nodes.

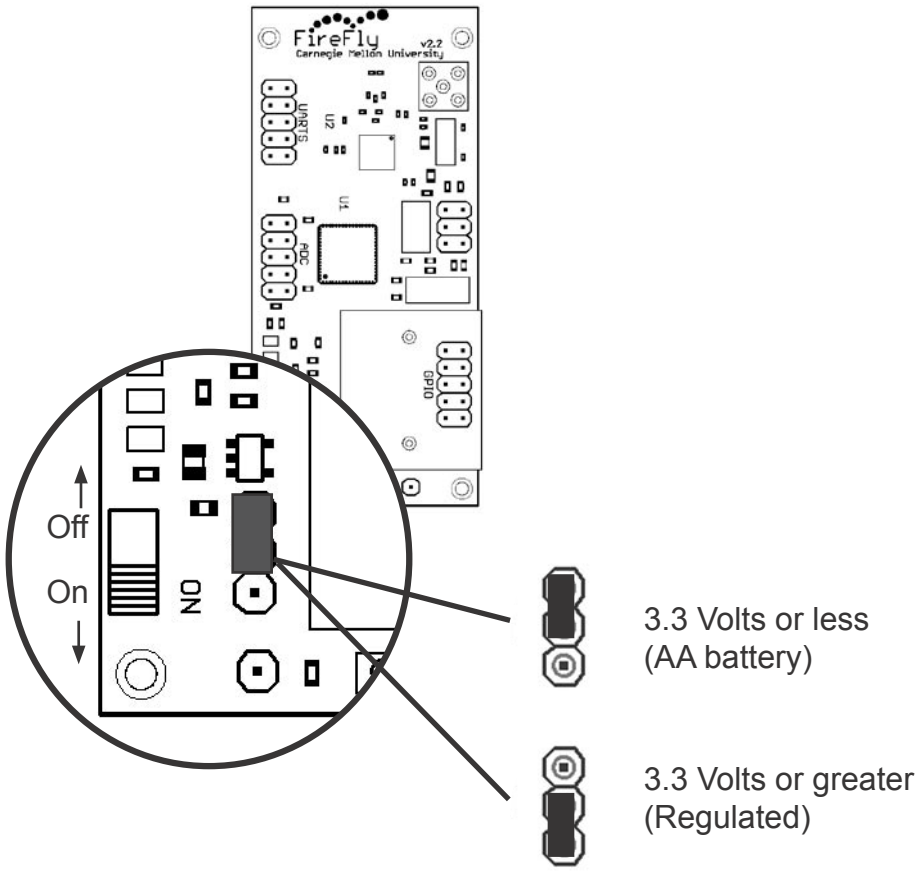


3. HARDWARE CONNECTIONS



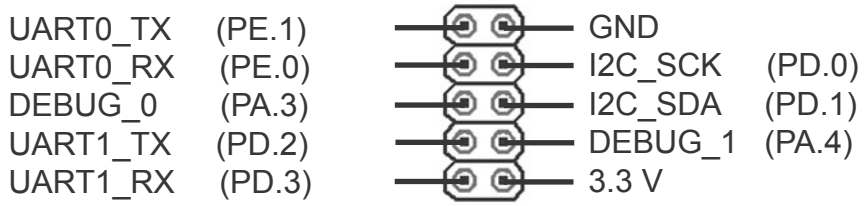
Power

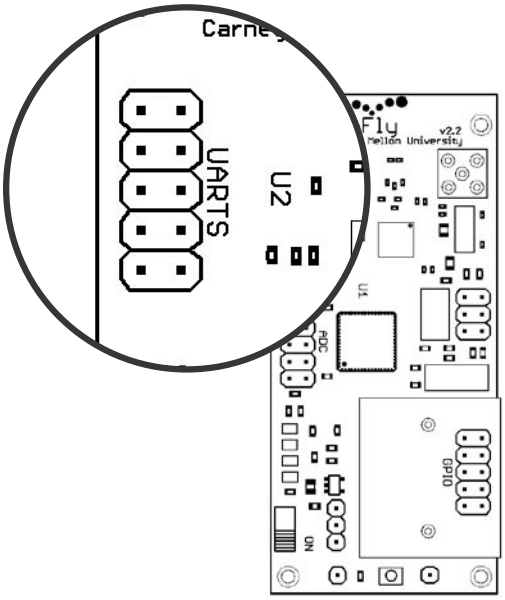
A FireFly node can be powered with or without a voltage regulator. Using a voltage regulator will allow the node to be powered from sources greater than 3.3 volts (up to 10 volts). However, when voltages at 3.3 volts or lower are available (for instance from two AA batteries), it is more efficient to operate the node directly from the power source rather than using the voltage regulator. Linear voltage regulators are typically only 80-90% efficient. The figure below shows the two possible power regulation settings.



Header 1 (UARTS, I2C, GPIO, ISP)

Header 1 exposes two uarts, the i2c bus, and two GPIO pins. UART0 on the ATmega1281 is required for programming, which is why the FireFly node uses UART1 for its default debugging output. UART0 can be used during operation after the node has been programmed. Both UART0 and UART1 use TTL levels that require level shifting before connecting to an RS232 port. The FireFly programmer board has level shifters, or alternatively can load two virtuak communicatoin ports over USB. The two GPIO pins as well as the i2c are used by default as debugging pins that connect to the four debugging terminals on the FireFly programmer board. Power and ground are available as a convenience and can either power external boards, or if given 3.3 volts or less can be used to power the FireFly board (this voltage input is after the voltage regulator).

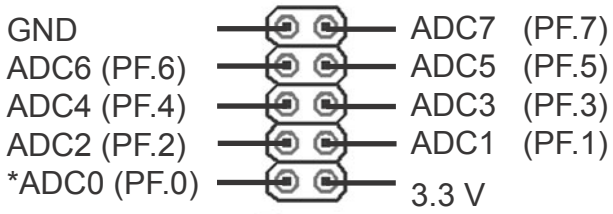


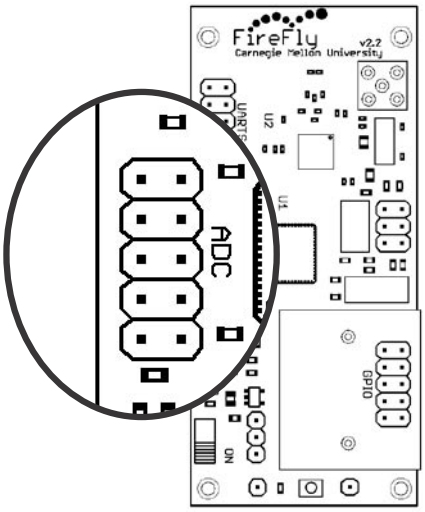


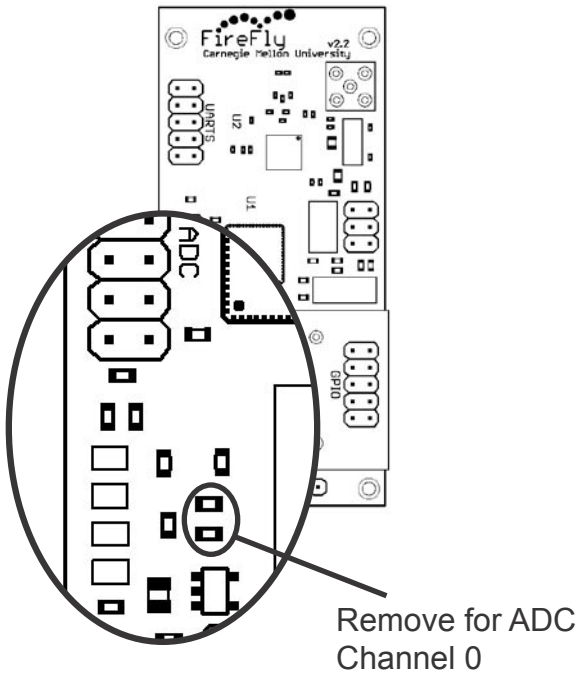
Header 2 (ADC)

Header 2 exposes power, ground and all eight analog-to-digital (ADC) ports available on the processor. This port is primarily intended for connecting sensors expansion devices.

ADC channel 0 is internally connect to a voltage divider that allows monitoring of external voltages greater than 3.3 volts. For voltages below 3.3 volts, the radio can be used to report voltage levels. Removing the two resistors in the voltage divider shown below will allow ADC 0 to be used as an ordinary input.



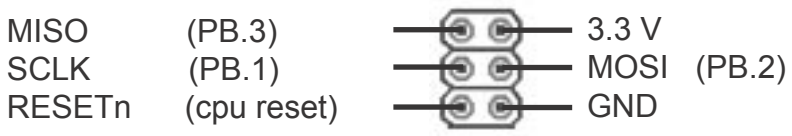


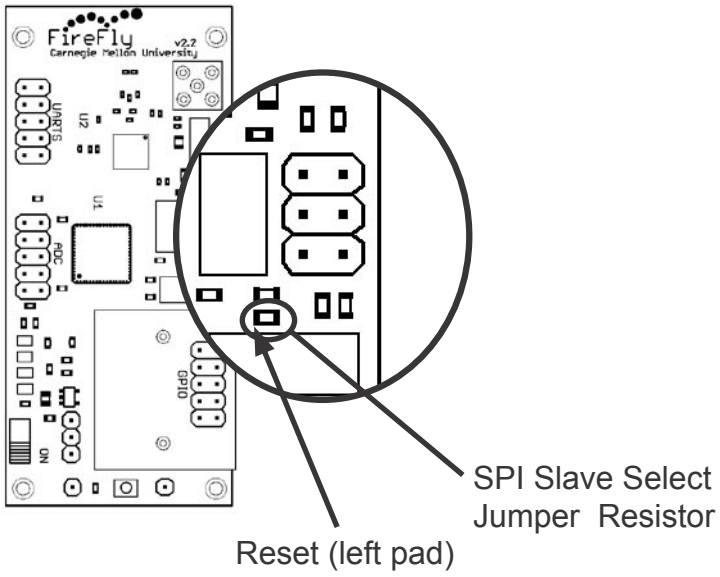


* Connected to Battery Level Voltage Divider

Header 3 (SPI / ISP)

The SPI / ISP header provides the interface for the in-system programming (ISP) board, as well as SPI master or slave devices. Typically the FireFly processor should be used as an SPI master since the CC2420 chip resides as a slave on the SPI bus. In certain situations, it may be possible to operate the node as an SPI slave. This requires removal of the SPI slave select jumper resistor. Normally, SPI slave select is connected to the reset pin of the processor so that the external programming board can reset the main CPU. Once this resistor is remove, the pin will instead be connected to the SPI slave select line. In order to program, the reset line from the programmer must now be connect to the left most pad where the resistor was removed.

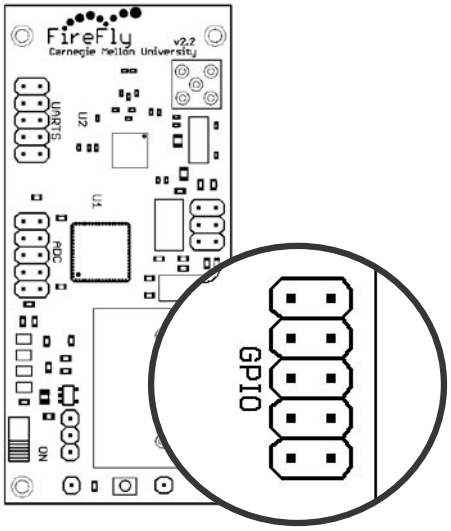




Header 4 (GPIO)

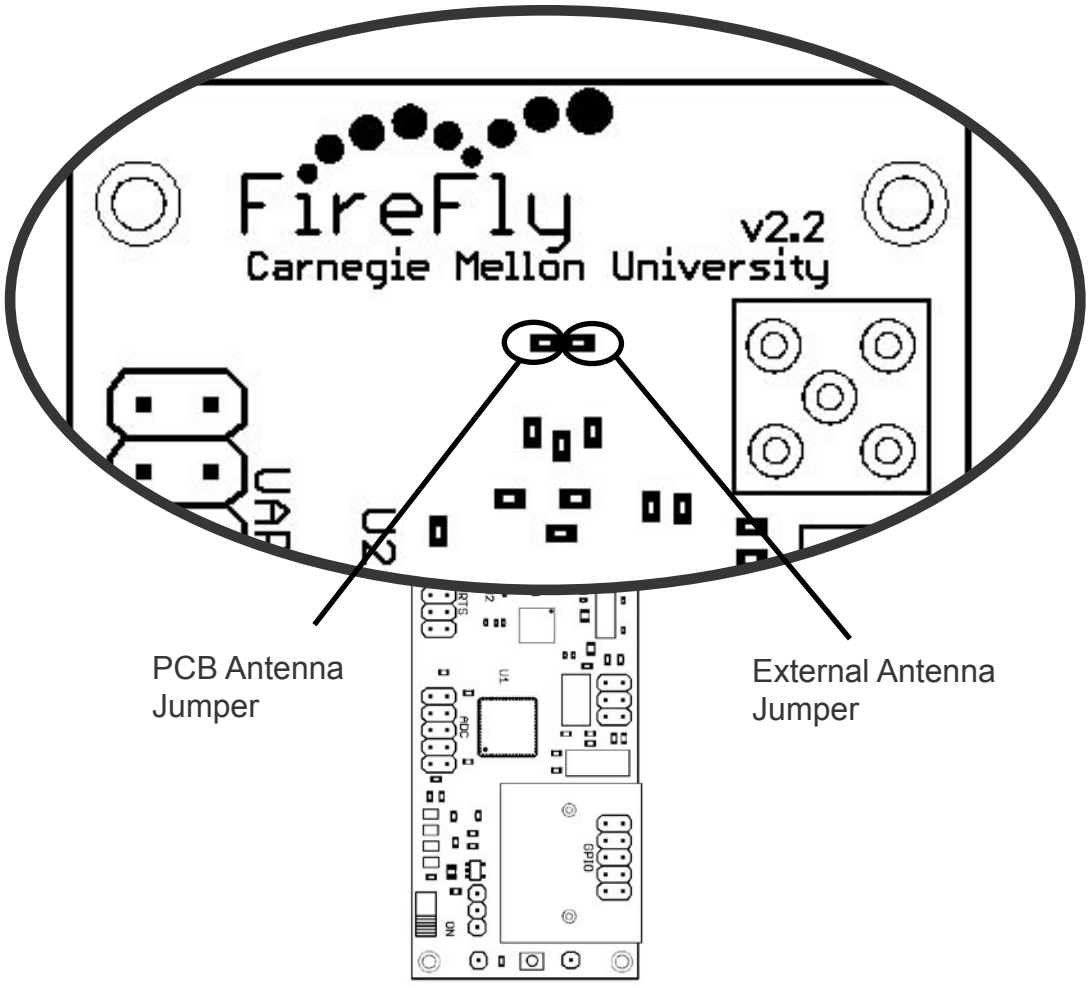
Header 4 is usually hidden under the mini-SD card slot. In cases when more GPIO is required it may be possible to connect to this header underneath the board, or by removing the mini-SD slot.

3.3V		GND
mmc_11	(PB.5)	n.c.
mmc_10	(PB.6)	n.c.
mmc_9	(PB.7)	n.c.
mmc_cs	(PE.6)	3.3 V



Antenna Configurations

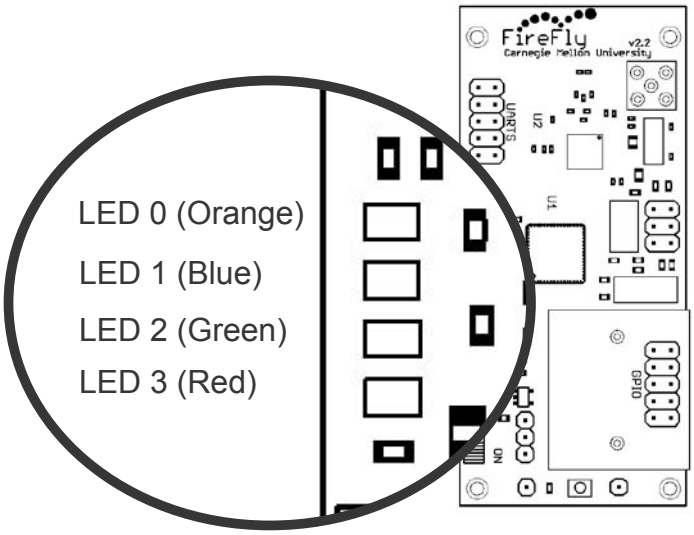
The FireFly node can operate using an internal Inverted-F style PCB antenna, or an external antenna connected to a reverser polarity SMA connector. Changing the position of the 5.6pF antenna selection capacitor shown below switches to the appropriate antenna.



LEDs

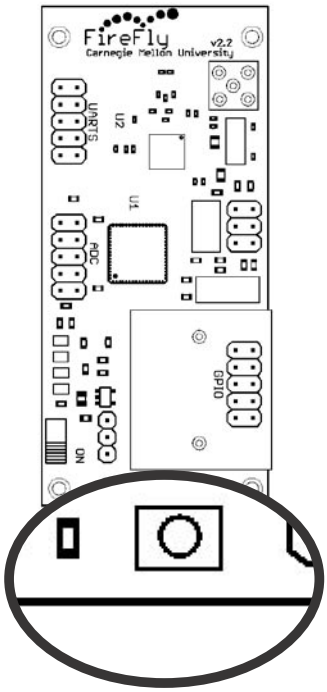
The LEDs are wired up as inverse logic. Pulling the pin low turns the LED on, while pulling the pin high disables the LED.

LED 0 - Port E.2
LED 1 - Port E.3
LED 2 - Port E.4
LED 3 - Port E.5



Button

The button is connected to Port A.7 and can be read as a digital input. Note, that you will likely have to debounce the button in software.



4. HARDWARE CHARACTERISTICS

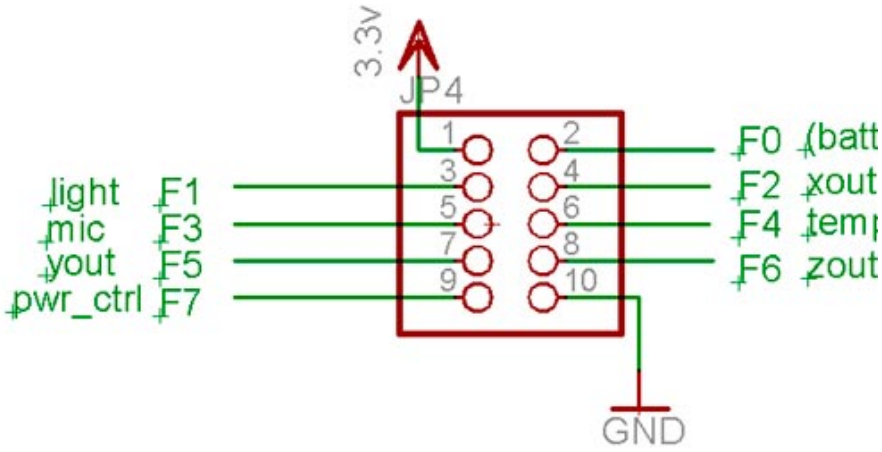
Power State	Current	Voltage
All Active	24.8 mA	3.0 V
CPU Idle + Radio Idle	0.20 uA	3.0 V
CPU Active	6 mA	3.0 V
Radio TX	17.4 mA	3.0 V
Radio RX	18.8 mA	3.0 V
MMC Card File Access	4mA	3.0 V

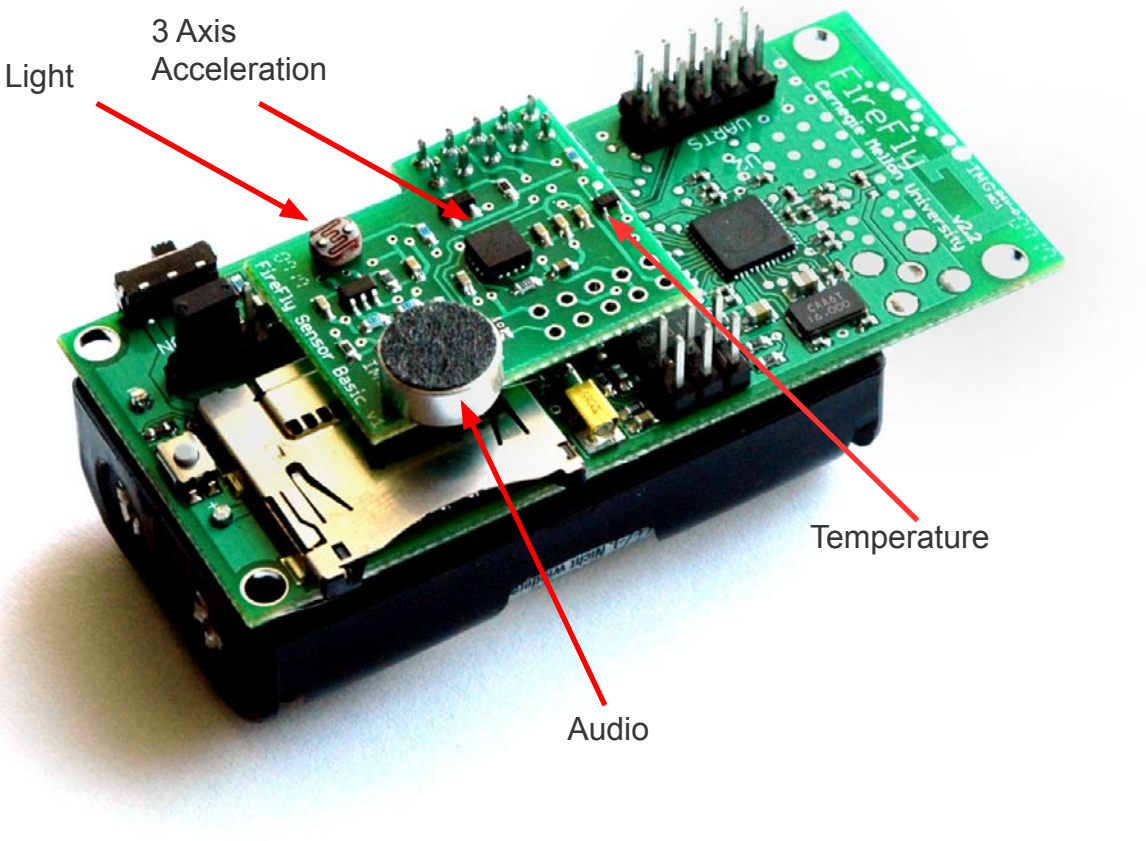
Table 1: Power consumption information for various components.

Component		Description
CPU		
	Type	ATmega1281
	RAM	8KB
	ROM	128KB (8KB taken by bootloader)
	frequency	(7.3728Mhz & 32.768KHz)
UART		
	Max Serial Out	230,400 bits / sec
	Max Serial In / Out	115,200 bits / sec
GPIO		
	Pin Low -> High Time	?
	Digital Pin Read	?
ADC		
	Resolution	10 bits
	Max Sampling Rate	15 KHz

Table 2: FireFly Node Characteristics

5. SENSORS


3.3V
JP4
1
2
3
4
5
6
7
8
9
10
light F1
mic F3
yout F5
pwr_ctrl F7
F0 (batt
F2 xout
F4 temp
F6 zout
GND

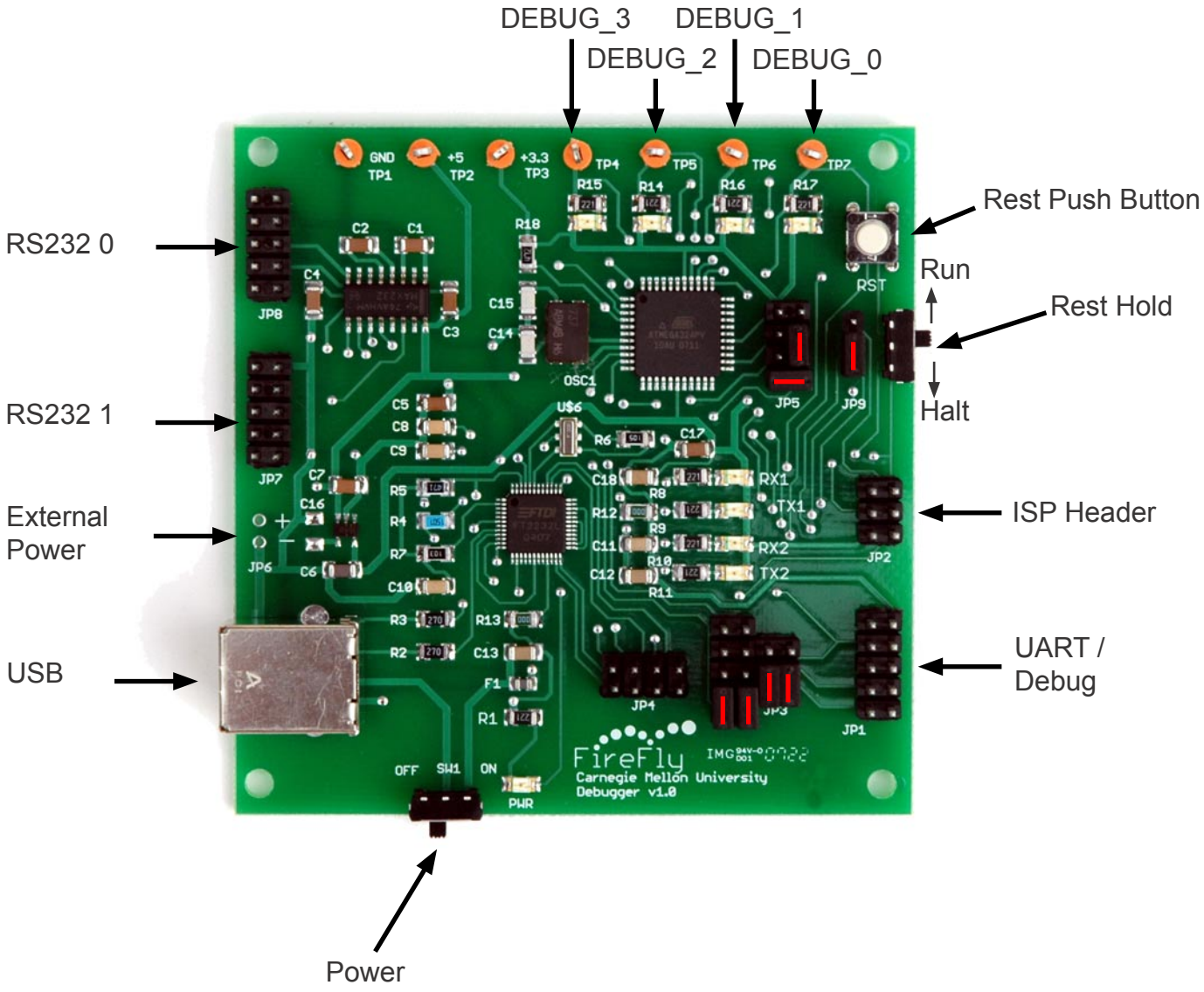

Light
3 Axis
Acceleration
Temperature
Audio

6. USING THE FIREFLY PROGRAMMER

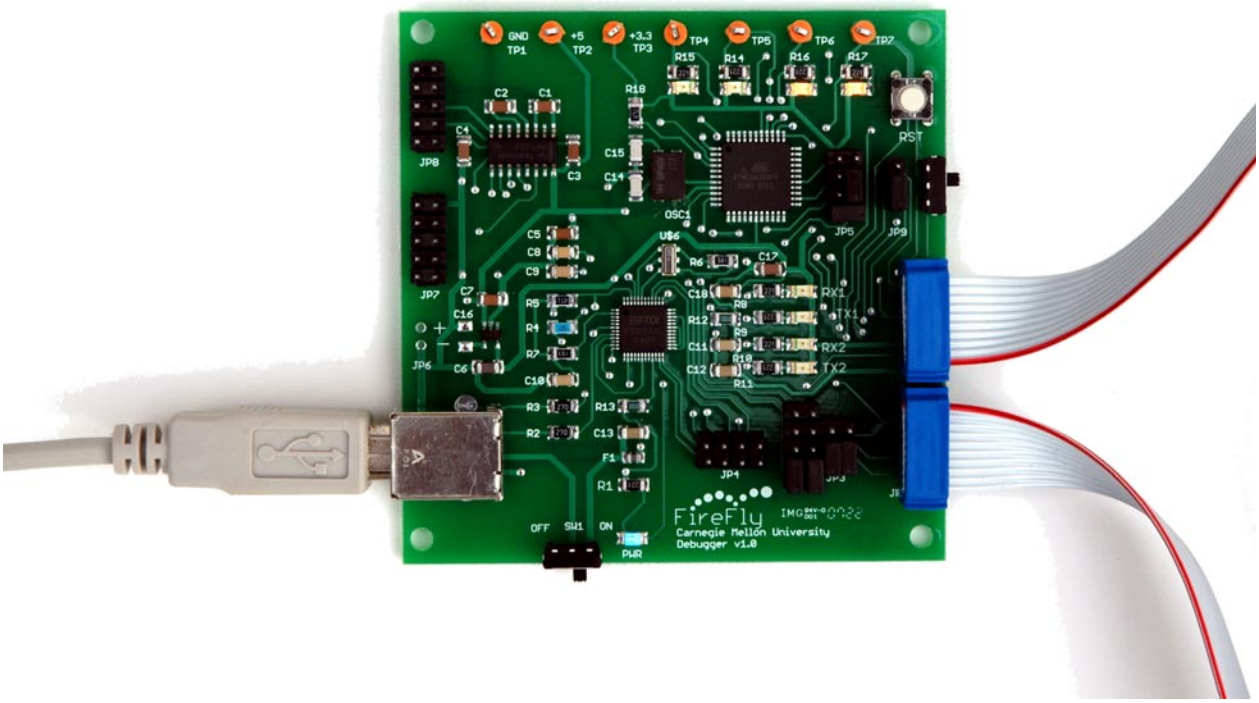
The FireFly debugging board can be used for programming nodes, sending / receiving serial data and timing of code section execution. The board will provide power to the target node from the connected computer's USB port. **DO NOT TURN ON THE NODE'S BATTERY POWER WHILE NODE IS CONNECTED TO THE PROGRAMMER!**

Connecting the Programmer

Before using the board, be sure that the jumpers (highlighted with red lines) are connected as shown below.



Next, connect the USB cable, and the two ribbon cables as shown below.



Switch on the programmer and confirm that the blue LED illuminates. Under linux, you can check to be sure that the device loaded correctly by typing: dmesg

```
usb 2-2: new full speed USB device using uhci_hcd and address 3
usb 2-2: new device found, idVendor=0403, idProduct=6010
usb 2-2: new device strings: Mfr=1, Product=2, SerialNumber=0
usb 2-2: Product: Dual RS232
usb 2-2: Manufacturer: FTDI
usb 2-2: configuration #1 chosen from 1 choice
ftdi_sio 2-2:1.0: FTDI USB Serial Device converter detected
drivers/usb/serial/ftdi_sio.c: Detected FT2232C
usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB0
ftdi_sio 2-2:1.1: FTDI USB Serial Device converter detected
drivers/usb/serial/ftdi_sio.c: Detected FT2232C
usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB1
```

This shows that two serial ports have been started at /dev/ttyUSB0 and /dev/ttyUSB1

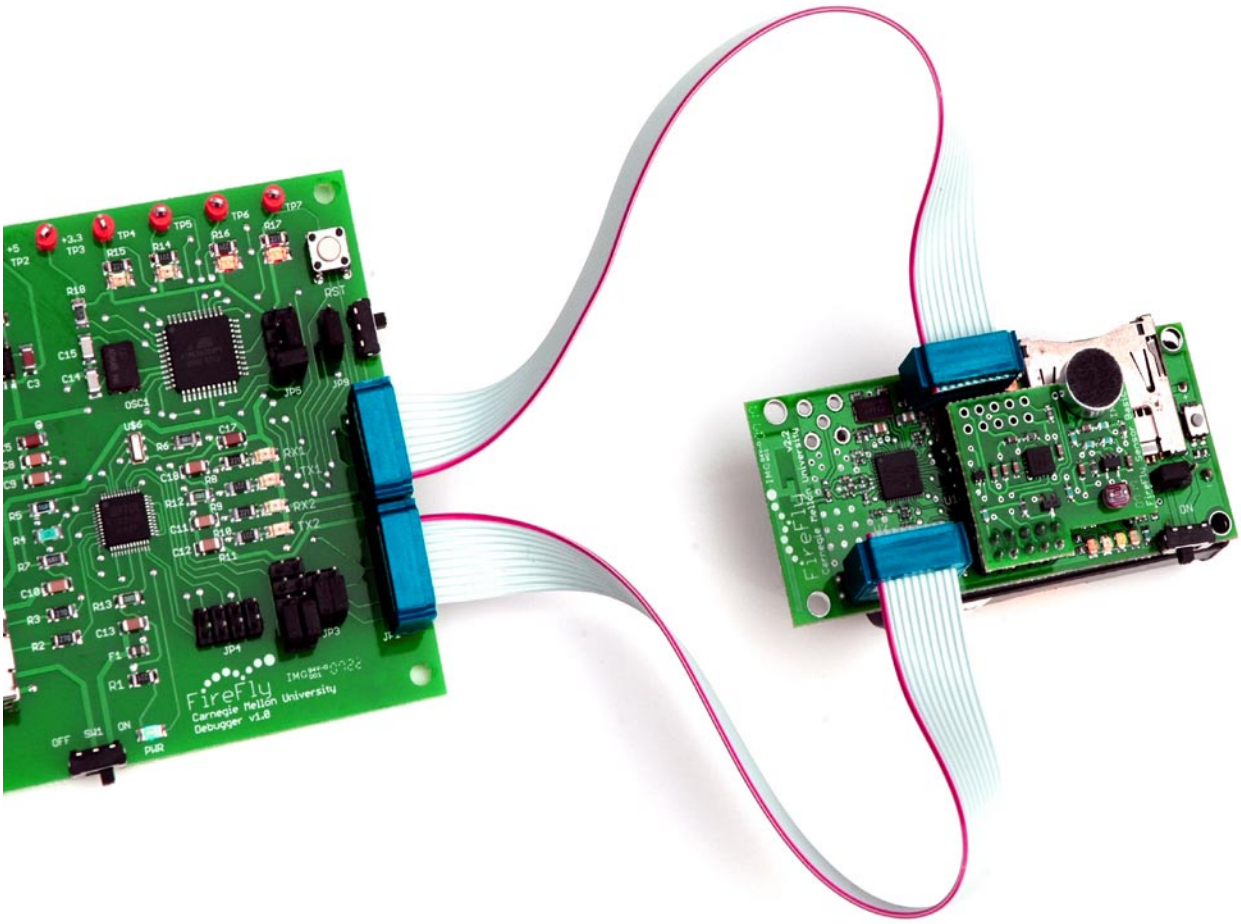
- The lower numbered port (/dev/ttyUSB0) is for debug messages
- The higher numbered port (/dev/ttyUSB1) is for the downloader and TimeScope

In windows, check in the device manager to see if the programming board loaded as two COM ports

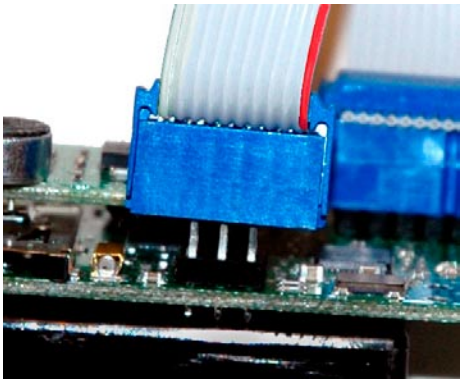
- The lower numbered port (COM1) is for debug messages
- The higher numbered port (COM2) is for the downloader

These port numbers will likely be different on different computers

Now connect the node as shown below. Be sure that the orienation of the cables is correct.



In case the 6 pin ribbon cable (programming cable) is not keyed, make sure the pins line up with the connection on the board.



Compiling Source

Once everything is installed, try compiling and downloading a project. Change Directories to the project you want to compile (for example basic_task)

```
cd nano-RK/projects/basic_task
```

Set the platform type in the makefile

```
# Platform name  cc2420DK, firefly2_2, micaZ
PLATFORM = firefly2_2
```

Compile the code

```
make clean
make
```

You can also set the NODE_ADDR #define that will be passed to your C code. If no value is provided, it defaults to 0.

```
Size after:
main.elf  :
section     size      addr
.data        194   8388864
.text      13918         0
.bss         992   8389058
.stab      36204         0
.stabstr   15067         0
Total      66375

Errors: none
Platform: firefly2
-------- end --------
```

Setting FireFly Fuses (first time only)

If this is the first time you have programmed a FireFly node, make sure the fuses are set correctly. Once the fuses are set, they should remain programmed unless there is a downloading problem later. The following avrdude commands set the FireFly fuses using the FireFly debugger (replace /dev/ttyUSB1 with the correct com port if different). Note that the -F forces the download for the atmega1281 using the atmega128 as the device code since many versions of avrdude will not recognize the atmega1281. You can execute the following script to set the fuses in the nano-RK/tools directory:

```
nano-RK/tools/ff-set-fuses
```

Which executes the following three lines:

```
avrdude -b115200 -F -p atmega128 -P /dev/ttyUSB1  -c avr910 -V -U lfuse:w:0xE6:m -v
avrdude -b115200 -F -p atmega128 -P /dev/ttyUSB1  -c avr910 -V -U hfuse:w:0xD9:m -v
avrdude -b115200 -F -p atmega128 -P /dev/ttyUSB1  -c avr910 -V -U efuse:w:0xFF:m -v
```

Downloading the Code

```
make program
```

You should see something like this if the code downloads correctly:

```
avrdude: Device signature = 0x1e9702
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file “main.hex”
avrdude: input file main.hex auto detected as Intel Hex
avrdude: writing flash (14108 bytes):

Writing | ################################################## | 100% 7.07s

avrdude: 14108 bytes of flash written
avrdude: safemode: Fuses OK
avrdude done.  Thank you.
```

If it takes longer than about 15 seconds to program, then the cables are probably not connected correctly.

Open up a terminal program like Minicom (see Minicom page for help)

Configure the program for:

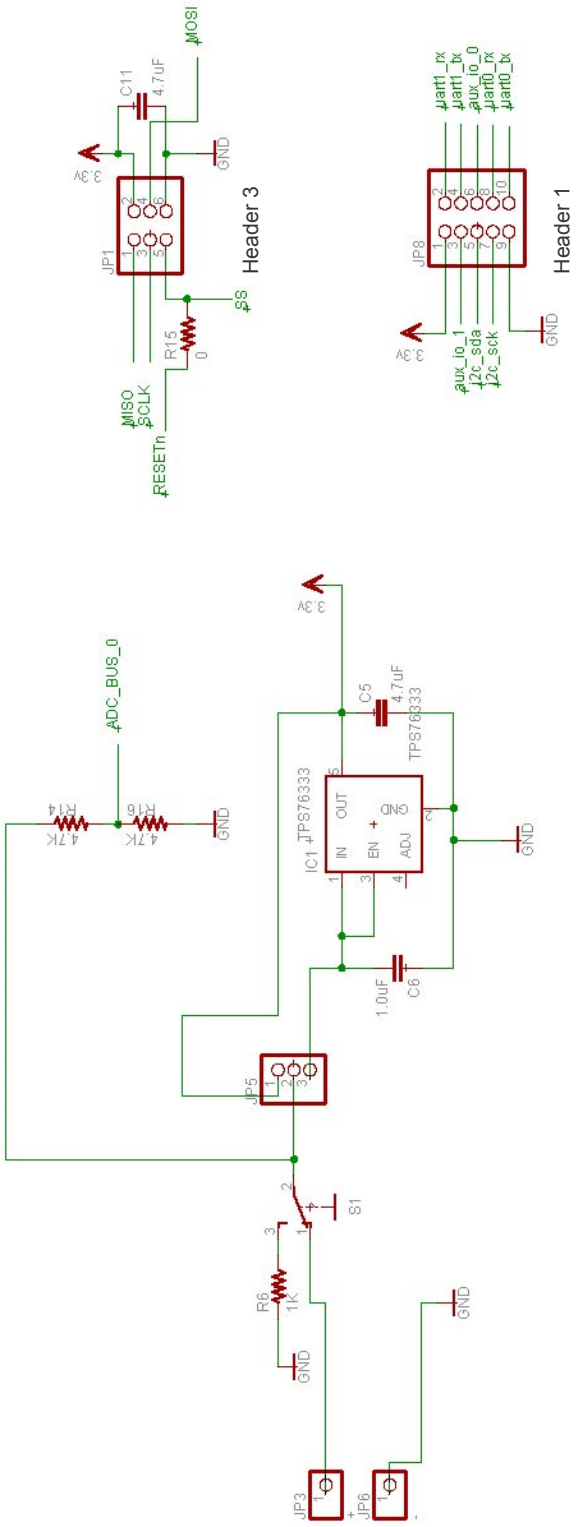
```
/dev/ttyUSB0 as the port path
Set the baudrate to 115200 8N1 with no Hardware / Software Flow Control
```

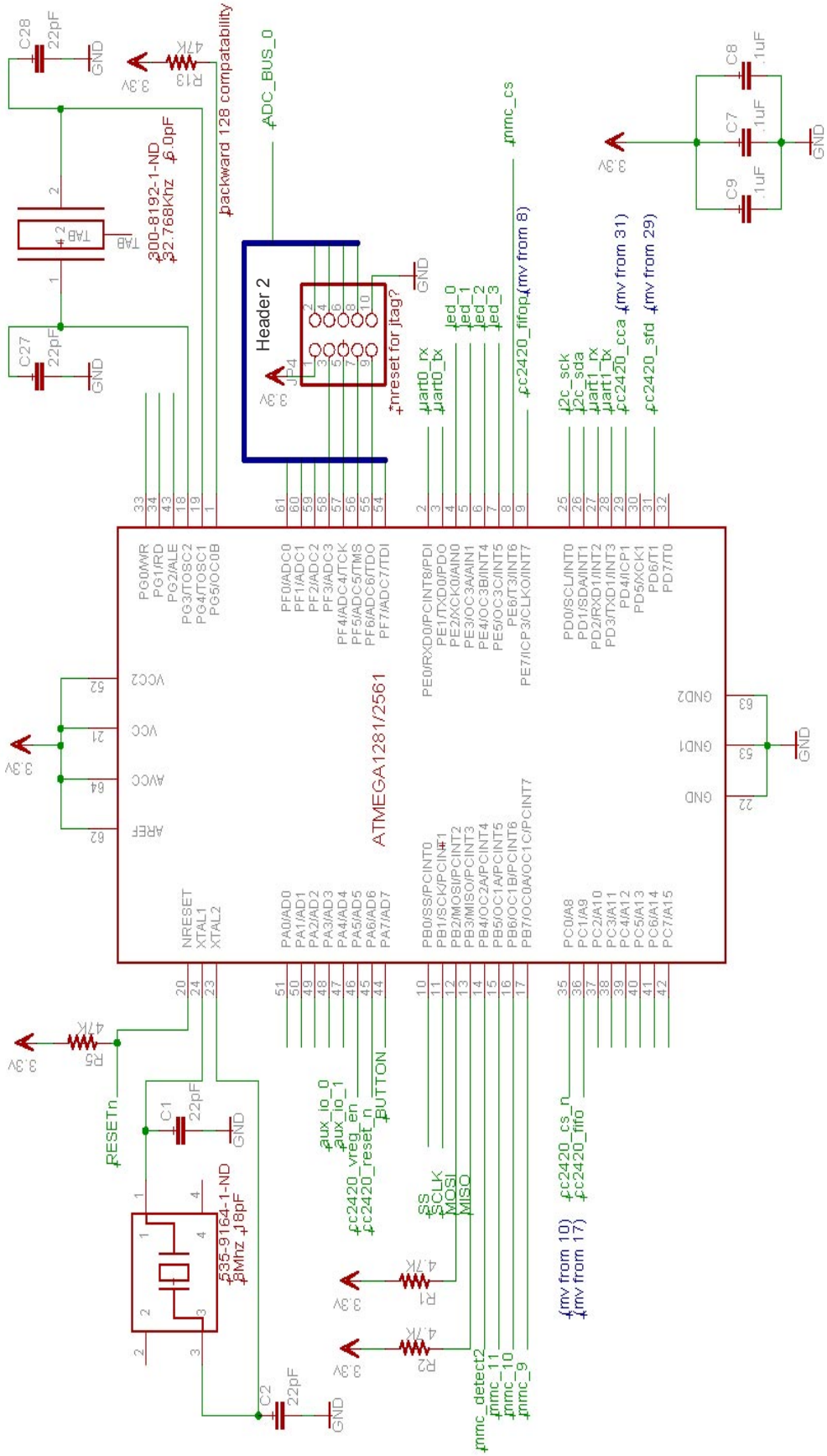
You should see something like this:

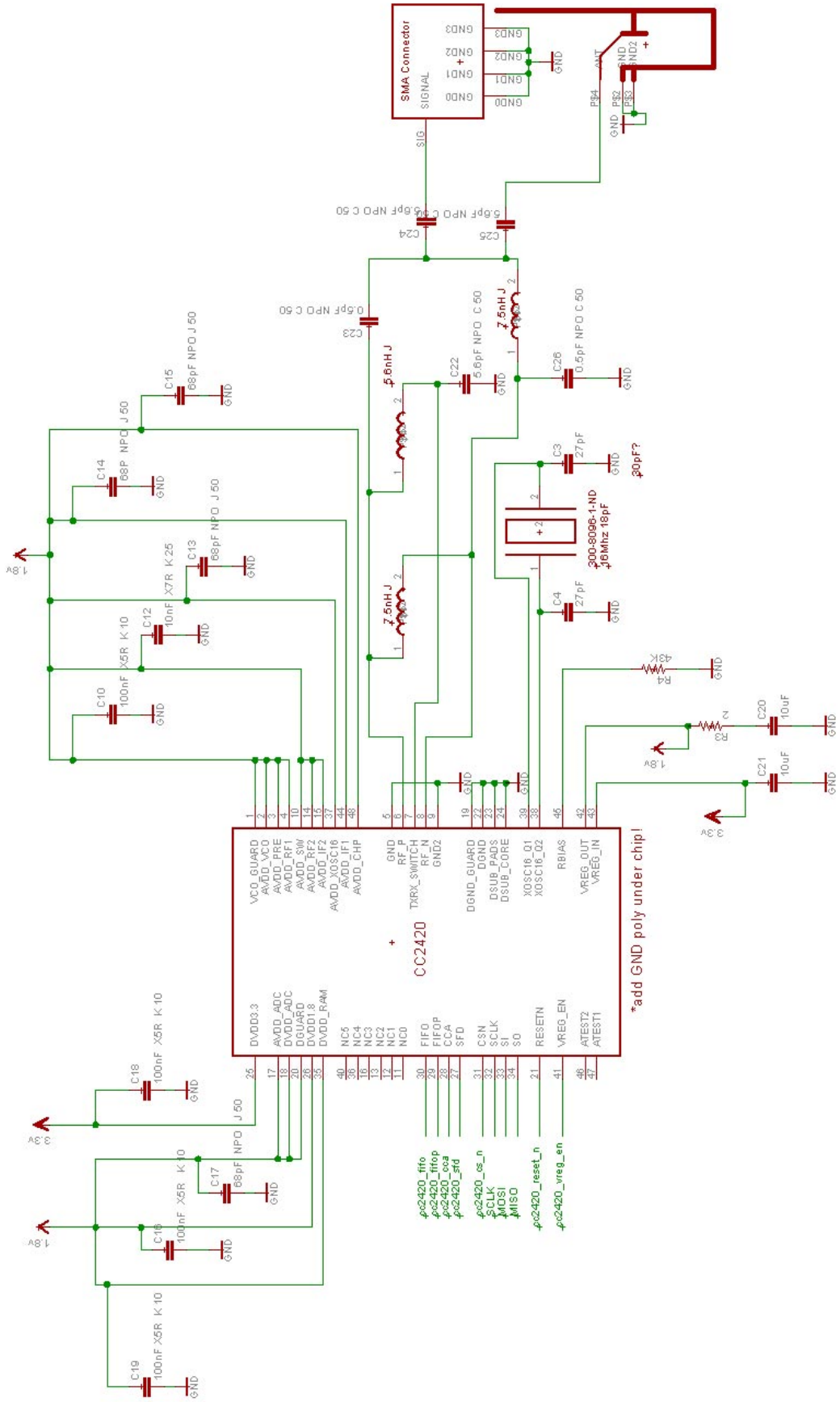
```
My node’s address is 5
Task1 PID=1
Task1 cnt=0
Task1 cnt=1
Task2 cnt=1
Task3 cnt=1
Task1 cnt=2
Task4 cnt=1
...
```

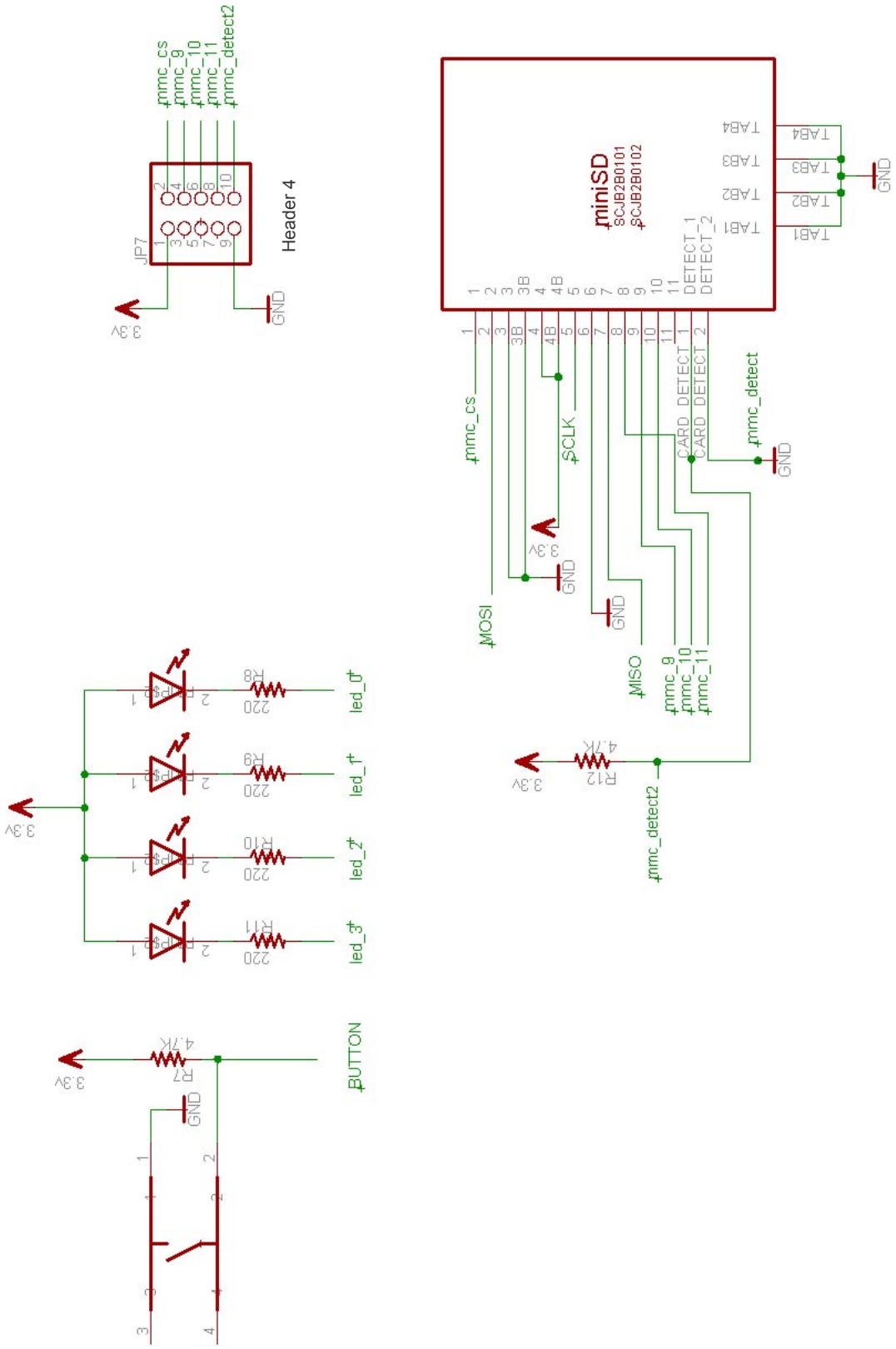
7. ASSEMBLY INFORMATION

Schematic

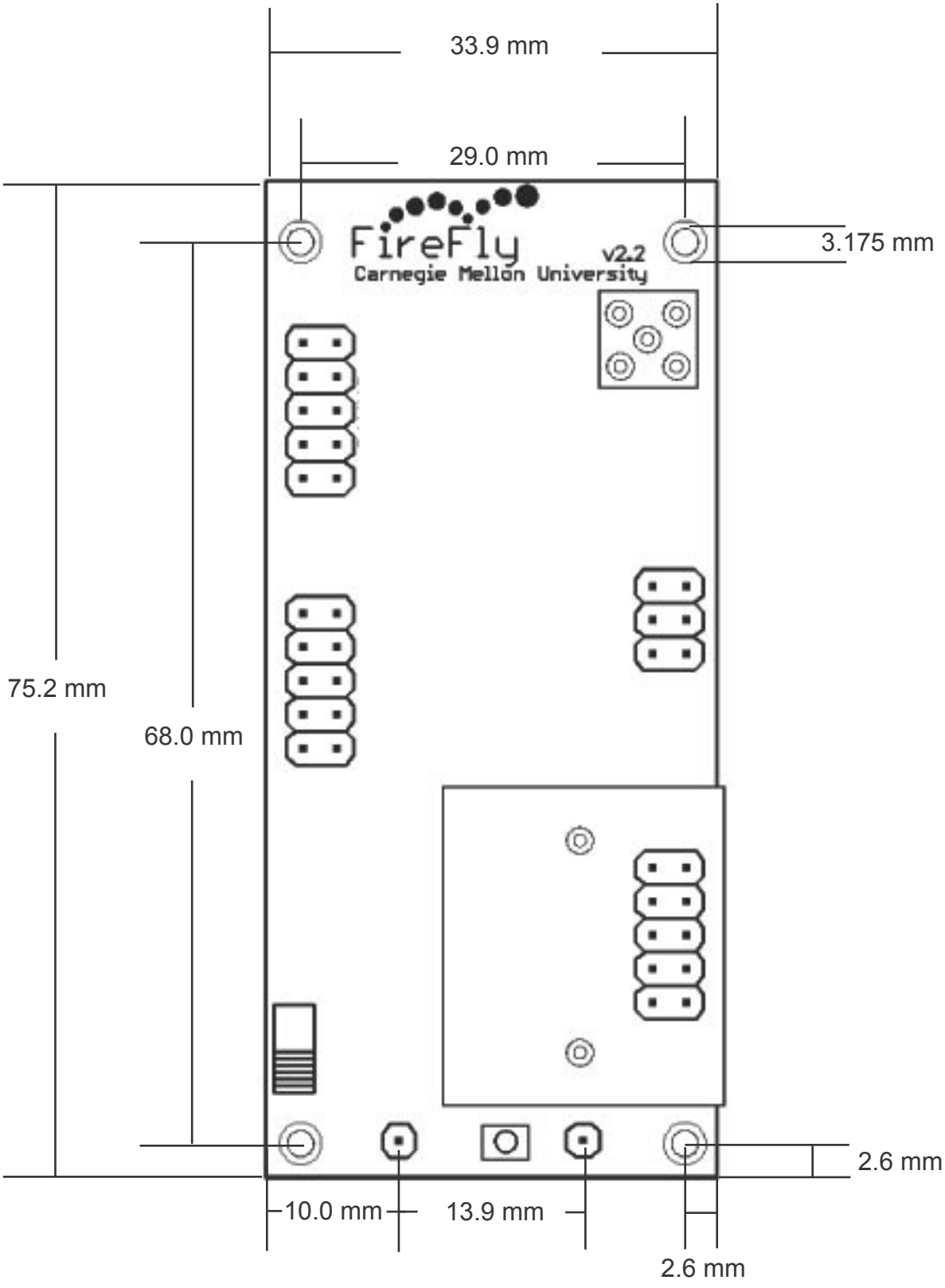



ATMEGA1281/2561
Header 2
JP4
C28
22pF
C27
22pF
C1
22pF
C2
22pF
C8
.1uF
C7
.1uF
C9
.1uF
R13
4.7K
R5
4.7K
R1
4.7K
R2
4.7K
TAB
300-8192-1-ND
32.768Khz
6.0pF
535-9164-1-ND
8Mhz
18pF
3.3V
GND
ADC_BUS_0
backward 128 compatability
nreset for jtag?
mmc_cs
uart0_rx
uart0_tx
led_0
led_1
led_2
led_3
c2420_fifop (mv from 8)
i2c_sck
i2c_sda
uart1_rx
uart1_tx
c2420_cca (mv from 31)
c2420_sfd (mv from 29)
RESETn
aux_io_0
aux_io_1
c2420_vreg_en
c2420_reset_n
BUTTON
SS
SCLK
MOSI
MISO
mmc_detect2
mmc_11
mmc_10
mmc_9
c2420_cs_n
c2420_fifo
(mv from 10)
(mv from 17)


SMA Connector
SIGNAL
CC2420
*add GND poly under chip!
300-8096-1-ND
16Mhz 18pF
30pF?
7.5nH J
5.6nH J
1.8v
3.3v


mmc_cs
mmc_9
mmc_10
mmc_11
mmc_detect2
JP7
Header 4
3.3v
GND
miniSD
SCJB2B0101
SCJB2B0102
TAB4
TAB3
TAB2
TAB1
DETECT_1
DETECT_2
3B
4B
SCLK
MOSI
MISO
CARD DETECT 1
CARD DETECT 2
mmc_detect
R8
R9
R10
R11
R12
220
4.7K
led_0
led_1
led_2
led_3
BUTTON
R7

Dimensions


33.9 mm
29.0 mm
3.175 mm
FireFly
v2.2
Carnegie Mellon University
75.2 mm
68.0 mm
2.6 mm
10.0 mm
13.9 mm
2.6 mm