softusbduino Documentation

Release 1.0.0

ponty

February 16, 2012

CONTENTS

softusbduino

Date February 16, 2012

PDF softusbduino.pdf

CHAPTER

ONE

ABOUT

softusbduino is a Python package and Arduino firmware library. They can be used together to control the Arduino board over USB in Python.

Links:

- home: https://github.com/ponty/softusbduino
- documentation: http://ponty.github.com/softusbduino

Hierarchy: Python Application -> softusbduino python library -> PyUSB -> libusb -> USB cable -> V-USB hardware -> Arduino -> V-USB library -> softusbduino firmware

Features:

- Possible usage: prototyping or creating simple low speed USB devices.
- firmware should be load only once to the Arduino board.
- 1 low level call takes 2 ms in tests
- **python library functions:**
  - read or write all registers
  - call arduino functions
  - read many defines (example: F_CPU)
- Python USB back-end: PyUSB 1.0 library
- Arduino USB back-end: V-USB library

Known problems:

- tested only on Linux + arduino 0022 + ATmega88 board
- pull-up read is not implemented
- PWM read is not implemented
- PWM config is hardcoded

similar projects:

- https://github.com/HashNuke/Python-Arduino-Prototyping-API
- http://code.google.com/p/vusb-for-arduino/

CHAPTER

TWO

BASIC USAGE OF PROTOTYPING

```
from softusbduino.protoapi import *

def setup():
    pinMode(13, OUTPUT);

def loop():
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);

sketch = Sketch(setup, loop)
sketch.run()
```

CHAPTER

THREE

INSTALLATION

3.1 General

- install Python
- install pip
- install arduino
- **install SoftUsb subdirectory as arduino library**
  - Manual installation: http://arduino.cc/en/Guide/Environment#libraries
  - **Automatic installation:**
    - install confduino
    - install the library: `python -m confduino.libinstall https://github.com/ponty/softusbduino/zipball/master`
- install python package:

  ```
  # as root
  pip install https://github.com/ponty/softusbduino/zipball/master
  ```

3.2 Ubuntu

```
sudo apt-get install arduino python-pip
sudo pip install confduino
sudo pip install https://github.com/ponty/softusbduino/zipball/master
sudo python -m confduino.libinstall https://github.com/ponty/softusbduino/zipball/master
# optional for examples
sudo pip install matplotlib traits traitsui
```

3.3 Upload firmware

1. start Arduino
2. open examples > SoftUsb > Simple
3. upload to board

CHAPTER

FOUR

USAGE

```
>>> from softusbduino import *
>>>
>>> # read defines
>>> board = Arduino()
>>>
>>> # reset pin directions
>>> board.reset()
>>>
>>> # constants in python library
>>> print '0x%X' % board.idVendor
0x16C0
>>> print '0x%X' % board.idProduct
0x5DF
>>> print board.bandgap_voltage
1.1
>>>
>>> # constants in firmware
>>> print board.usbMinusPin
4
>>> print board.usbPlusPin
2
>>> print board.pinCount
20
>>> print board.pinRange()
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> print board.pinRange('digital')
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
>>> print board.pinRange('analog')
[14, 15, 16, 17, 18, 19]
>>>
>>> # supply voltage
>>> print board.vcc
4.85517241379
>>> print board.u_vcc
4.85517241379+/-0.0418549346017
>>>
>>> # pin
>>> print board.pin(8).nr
8
>>> print board.pin('D8').nr
8
>>> print board.pin('A2').nr
16
>>> print board.pin('D13').programming_function
SCK
>>>
>>> # pin mode
>>> board.pinMode(8, OUTPUT)
>>> print board.readPinMode(8)
1
>>> print board.pin(’D8’).mode
1
>>> board.pin(’D8’).mode = INPUT
>>> print board.readPinMode(8)
0
>>> print board.pin(’D8’).mode
0
>>>
>>> # analog read
>>> print board.pin(’A2’).analogRead()
26
>>> print board.pin(’A2’).an_in
0
>>> print board.pin(’A2’).u_an_in
0.0+/-2.0
>>>
>>> # digital read
>>> print board.pin(’D8’).dig_in
0
>>>
>>> # pullup
>>> pinD8 = board.pin(’D8’)
>>> pinD8.pullup = True
>>> print pinD8.pullup
True
>>>
>>> # digital write
>>> board.pin(’D8’).dig_out = 1
>>> print board.pin(’D8’).dig_out
1
>>> board.pin(’D8’).dig_out = 0
>>> print board.pin(’D8’).dig_out
0
>>>
>>> # PWM
>>> print board.pin(’D9’).pwm_available
True
>>> print board.pin(’D9’).timer_register_name
TCCR1B
>>> print board.pin(’D9’).pwm_frequencies_available
[39062.5, 4882.8125, 610.3515625, 152.587890625, 38.14697265625]
>>> print board.pin(’D9’).pwm_frequency
610.3515625
>>> print board.pin(’D9’).divisors_available
[1, 8, 64, 256, 1024]
>>> print board.pin(’D9’).divisor
64
>>> board.pin(’D9’).divisor = 256
>>> print board.pin(’D9’).pwm_frequency
152.587890625
>>> print board.pin(’D9’).divisor
256
>>> board.pin(’D9’).pwm_frequency = 38
>>> print board.pin(’D9’).pwm_frequency
38.1469726562
>>> print board.pin(’D9’).divisor
1024
>>> board.pin(’D9’).pwm_out = 4
>>>
>>> # read defines
>>> print board.defines.MCU_DEFINED
```

```
__AVR_ATmega88__
>>> print board.defines.F_CPU
20000000
>>> print board.defines.__DATE__
Feb 14 2012
>>> print board.defines.MOSI
11
>>> print board.defines.USB_CFG_DMINUS_BIT
4
>>> print board.defines.ARDUINO
22
>>> print board.defines.__AVR_LIBC_VERSION__
10701
>>> print board.defines.A0
14
>>>
>>> # read/write register
>>> board.registers.DDRB = 0
>>> print board.registers.DDRB
0
>>> print board.pin(8).mode
0
>>> board.registers.DDRB = 1
>>> print board.registers.DDRB
1
>>> print board.pin(8).mode
1
>>> board.pin(8).mode = INPUT
>>> print board.registers.DDRB
0
>>> print board.pin(8).mode
0
>>>
>>>
>>> board.reset()
```

4.1 Code generation

Integer defines should be listed in softusbduino/intdefs.csv. String defines are hardcoded. Registers and MCU names are read from AVR Libc directory (/usr/lib/avr/include/avr/).

Run codegen.py to update generated files:

- softusbduino/generated_registers.csv
- SoftUsb/generated_registers.h
- SoftUsb/generated_intdefs.h
- SoftUsb/generated_mcu.h
- SoftUsb/generated_version.h

CHAPTER

FIVE

EXAMPLES

5.1 Simple example

```
from entrypoint2 import entrypoint
from softusbduino.arduino import Arduino


@entrypoint
def main():
    board = Arduino()
    print board.defines.F_CPU
```

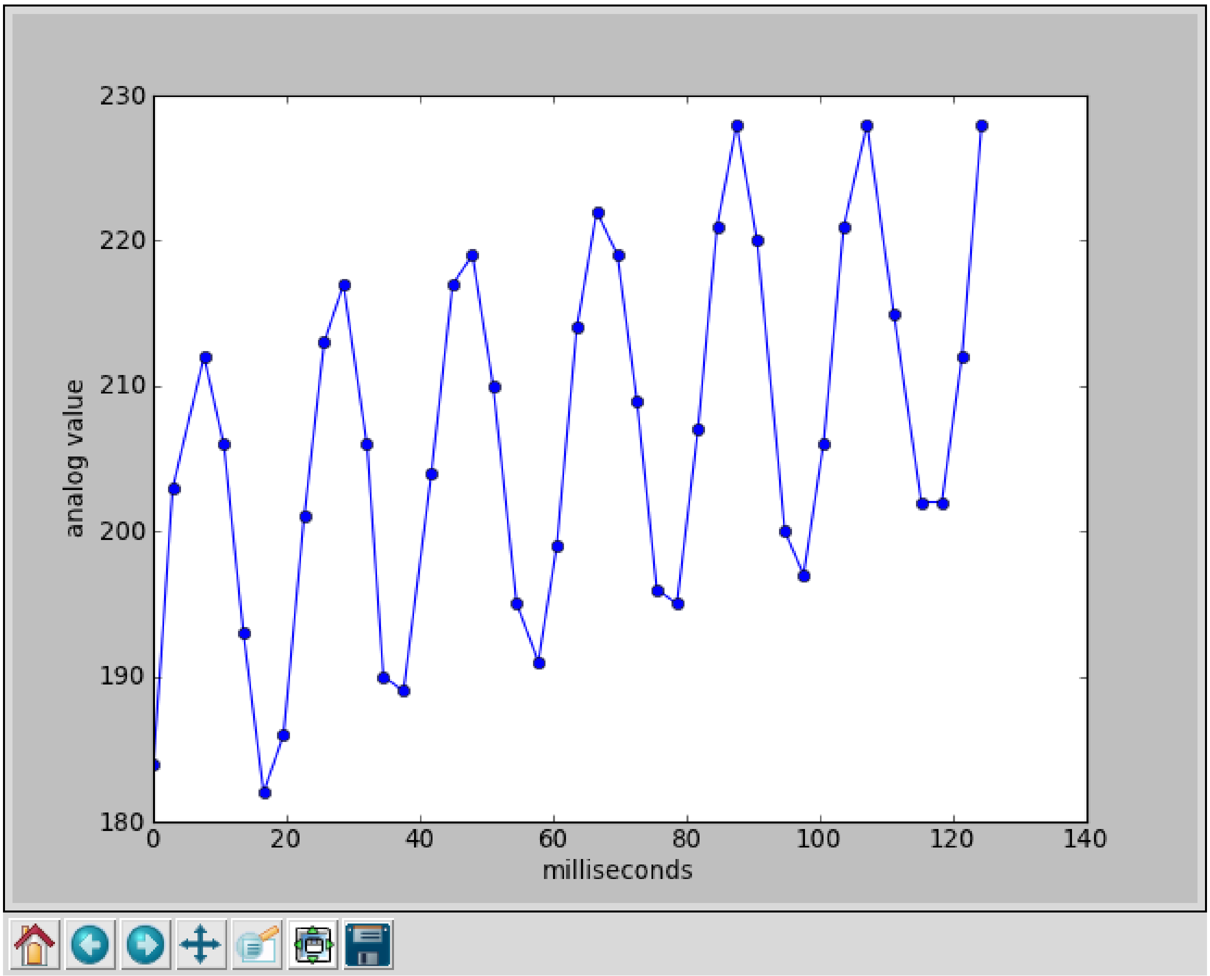
```
$ python -m softusbduino.examples.simple
20000000
```

5.2 Plot

```
from entrypoint2 import entrypoint
from matplotlib.ticker import FuncFormatter
from softusbduino.arduino import Arduino
import matplotlib.pyplot as plt
import time

@entrypoint
def main(n=40, pin_nr=13, reset=False):
    '''
    measuring analog input
    '''
    board = Arduino(reset=reset)

    x = []
    y = []
    start = time.time()
    for i in range(n):
        t = time.time() - start
        v = board.analogRead(pin_nr)
        x.append(t)
        y.append(v)
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(x, y, 'b-o')

    ax.yaxis.set_major_formatter(FuncFormatter(lambda x, pos: ('%d') % (x)))
    ax.set_ylabel('analog value')
    ax.xaxis.set_major_formatter(FuncFormatter(lambda x, pos: '%.0f' % (1000 * x)))
    ax.set_xlabel('milliseconds')
    plt.show()
```

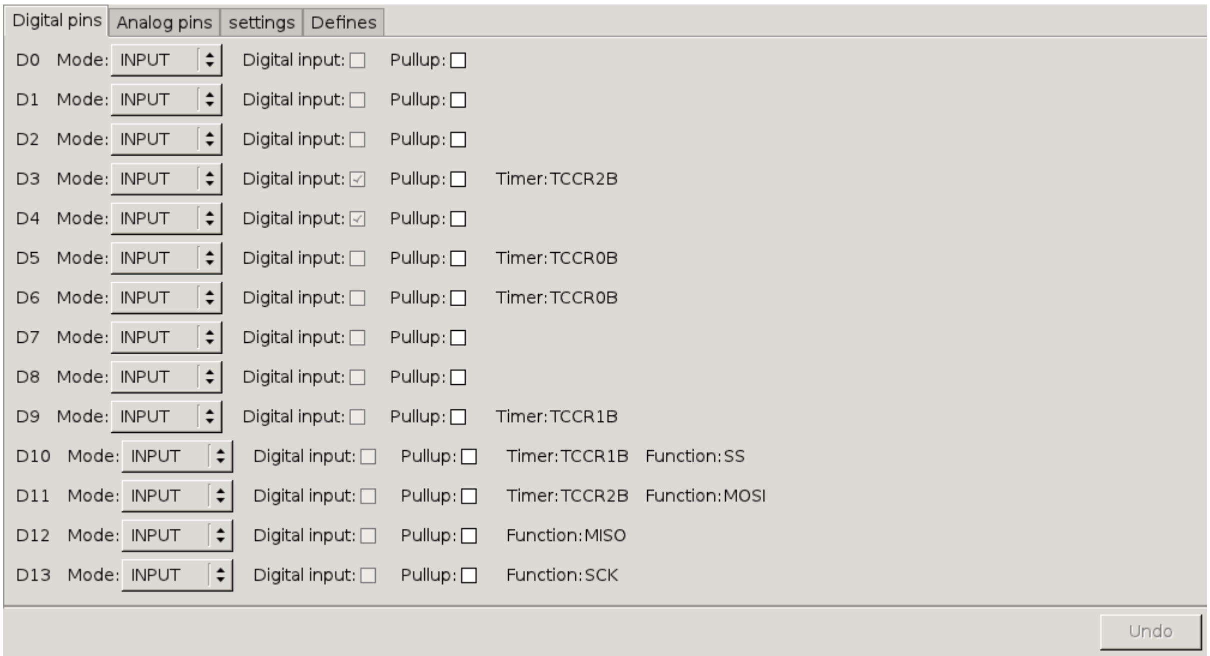
```
$ python -m softusbduino.examples.analogplot --help
usage: analogplot.py [-h] [--n N] [-p PIN_NR] [-r] [--debug]

measuring analog input

optional arguments:
  -h, --help            show this help message and exit
  --n N
  -p PIN_NR, --pin-nr PIN_NR
  -r, --reset
  --debug               set logging level to DEBUG
```

```
$ python -m softusbduino.examples.analogplot
```



5.3 Demo GUI

```
$ python -m softusbduino.examples.guidemo
```



D0 Mode: INPUT Digital input: □ Pullup: □
D1 Mode: INPUT Digital input: □ Pullup: □
D2 Mode: INPUT Digital input: □ Pullup: □
D3 Mode: INPUT Digital input: ☑ Pullup: □ Timer: TCCR2B
D4 Mode: INPUT Digital input: ☑ Pullup: □
D5 Mode: INPUT Digital input: □ Pullup: □ Timer: TCCR0B
D6 Mode: INPUT Digital input: □ Pullup: □ Timer: TCCR0B
D7 Mode: INPUT Digital input: □ Pullup: □
D8 Mode: INPUT Digital input: □ Pullup: □
D9 Mode: INPUT Digital input: □ Pullup: □ Timer: TCCR1B
D10 Mode: INPUT Digital input: □ Pullup: □ Timer: TCCR1B Function: SS
D11 Mode: INPUT Digital input: □ Pullup: □ Timer: TCCR2B Function: MOSI
D12 Mode: INPUT Digital input: □ Pullup: □ Function: MISO
D13 Mode: INPUT Digital input: □ Pullup: □ Function: SCK

5.4 prototyping

softusbduino/examples/proto/Blink.py

```
'''
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Converted from Arduino example.
'''

from softusbduino.protoapi import *

def setup():
    pinMode(13, OUTPUT);

def loop():
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);



sketch = Sketch(setup, loop)
sketch.run()
```

softusbduino/examples/proto/AnalogInOutSerial.py

```
'''
  Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the serial monitor.

 The circuit:
 * potentiometer connected to analog pin 0.
```

```
   Center pin of the potentiometer goes to the analog pin.
   side pins of the potentiometer go to +5V and ground
 * LED connected from digital pin 9 to ground

  Converted from Arduino example.
'''

from softusbduino.protoapi import *

# These constants won't change.  They're used to give names
# to the pins used:
analogInPin = A0;  # Analog input pin that the potentiometer is attached to
analogOutPin = 9; # Analog output pin that the LED is attached to

sensorValue = 0;        # value read from the pot
outputValue = 0;        # value output to the PWM (analog out)

def setup():
    # initialize serial communications at 9600 bps:
    Serial_begin(9600);


def loop():
    # read the analog in value:
    sensorValue = analogRead(analogInPin);
    # map it to the range of the analog out:
    outputValue = map(sensorValue, 0, 1023, 0, 255);
    # change the analog out value:
    analogWrite(analogOutPin, outputValue);

    # print the results to the serial monitor:
    Serial_print("sensor = " );
    Serial_print(sensorValue);
    Serial_print("\t output = ");
    Serial_println(outputValue);

    # wait 10 milliseconds before the next loop
    # for the analog-to-digital converter to settle
    # after the last reading:
    delay(10);

sketch = Sketch(setup, loop)
sketch.run()
```

CHAPTER

SIX

TESTS

Test system versions:

```
$ python -m softusbduino.lsversion
platform        Linux-3.0.0-12-generic-i686-athlon-with-LinuxMint-12-lisa
python          2.7.2+
```

Performance test:

```
$ python -m softusbduino.examples.performance
performance test
n= 100

analogRead(0)                           3.98 ms per call,   251 call per second
pinMode(8,0)                            3.82 ms per call,   262 call per second
digitalRead(8)                          3.70 ms per call,   270 call per second
digitalPinToBitMask(0)                  0.05 ms per call, 18292 call per second
digitalPinToPort(0)                     0.12 ms per call,  8278 call per second
portModeRegister(0)                     3.23 ms per call,   310 call per second
defines.read_define("__TIME__")         0.09 ms per call, 10832 call per second
defines.read_define("MCU_DEFINED")      0.12 ms per call,  8401 call per second
readPinMode(0)                          3.83 ms per call,   261 call per second
vcc                                    22.96 ms per call,    44 call per second
pinCount                                0.60 ms per call,  1655 call per second
usbMinusPin                             0.79 ms per call,  1272 call per second
usbPlusPin                              0.76 ms per call,  1313 call per second
firmware_test()                         0.06 ms per call, 16098 call per second
defines.__TIME__                        0.06 ms per call, 15704 call per second
pin("A0").an_in                         3.46 ms per call,   289 call per second
reset()                               122.84 ms per call,     8 call per second
```

Dump registers and defines:

```
$ python -m softusbduino.examples.dump

=========================
attributes:
=========================
Rout            =              15
adc_accuracy    =               2
analog_range    =       (0, 1023)
bandgap_voltage =             1.1
idProduct       =            1503
idVendor        =            5824
manufacturer    =        obdev.at
pinCount        =              20
productName     =       LEDCtlHID
u_vcc           = 4.85517241379+/-0.0418549346017
usbMinusPin     =               4
usbPlusPin      =               2
```

```
vcc             =   4.85517241379


=========================
defines:
=========================
A0                   =                  14
ARDUINO              =                  22
E2END                =                 511
E2PAGESIZE           =                   4
FLASHEND             =                8191
F_CPU                =            20000000
MAGIC_NUMBER         =                  42
MCU_DEFINED          =    __AVR_ATmega88__
MISO                 =                  12
MOSI                 =                  11
RAMEND               =                1279
SCK                  =                  13
SOFTUSBDUINO_VERSION =               10000
SPM_PAGESIZE         =                  64
SS                   =                  10
USBDRV_VERSION       =            20100715
USB_CFG_DMINUS_BIT   =                   4
USB_CFG_DPLUS_BIT    =                   2
USB_CFG_IOPORT       =                   4
XRAMEND              =                1279
__AVR_LIBC_DATE_     =            20110216
__AVR_LIBC_VERSION__ =               10701
__DATE__             =         Feb 14 2012
__TIME__             =            19:29:51

=========================
registers:
=========================
ACSR                 = 0x30 @0x50
ADCH                 = 0x00 @0x79
ADCL                 = 0xE8 @0x78
ADCSRA               = 0x97 @0x7A
ADCSRB               = 0x00 @0x7B
ADMUX                = 0x4E @0x7C
ASSR                 = 0x00 @0xB6
CLKPR                = 0x00 @0x61
DDRB                 = 0x00 @0x24
DDRC                 = 0x00 @0x27
DDRD                 = 0x00 @0x2A
DIDR0                = 0x00 @0x7E
DIDR1                = 0x00 @0x7F
EEAR                 = 0x00 @0x41
EEARH                = 0x01 @0x42
EEARL                = 0x00 @0x41
EECR                 = 0x00 @0x3F
EEDR                 = 0x00 @0x40
EICRA                = 0x02 @0x69
EIFR                 = 0x00 @0x3C
EIMSK                = 0x01 @0x3D
GPIOR0               = 0x00 @0x3E
GPIOR1               = 0x00 @0x4A
GPIOR2               = 0x00 @0x4B
GTCCR                = 0x00 @0x43
ICR1                 = 0x18 @0x86
ICR1H                = 0x00 @0x87
ICR1L                = 0x18 @0x86
MCUCR                = 0x00 @0x55
```

```
MCUSR                = 0x01 @0x54
MONDR                = 0x35 @0x51
OCR0A                = 0x00 @0x47
OCR0B                = 0x00 @0x48
OCR1A                = 0x0B @0x88
OCR1AH               = 0x00 @0x89
OCR1AL               = 0x0B @0x88
OCR1B                = 0x00 @0x8A
OCR1BH               = 0x00 @0x8B
OCR1BL               = 0x00 @0x8A
OCR2A                = 0x00 @0xB3
OCR2B                = 0x00 @0xB4
OSCCAL               = 0x98 @0x66
PCICR                = 0x00 @0x68
PCIFR                = 0x00 @0x3B
PCMSK0               = 0x00 @0x6B
PCMSK1               = 0x00 @0x6C
PCMSK2               = 0x00 @0x6D
PINB                 = 0x00 @0x23
PINC                 = 0x00 @0x26
PIND                 = 0x18 @0x29
PORTB                = 0x00 @0x25
PORTC                = 0x00 @0x28
PORTD                = 0x00 @0x2B
PRR                  = 0x00 @0x64
SMCR                 = 0x00 @0x53
SP                   = 0xED @0x5D
SPCR                 = 0x00 @0x4C
SPDR                 = 0x00 @0x4E
SPH                  = 0x04 @0x5E
SPL                  = 0xED @0x5D
SPMCSR               = 0x00 @0x57
SPSR                 = 0x00 @0x4D
SREG                 = 0x82 @0x5F
TCCR0A               = 0x03 @0x44
TCCR0B               = 0x03 @0x45
TCCR1A               = 0x01 @0x80
TCCR1B               = 0x03 @0x81
TCCR1C               = 0x00 @0x82
TCCR2A               = 0x01 @0xB0
TCCR2B               = 0x04 @0xB1
TCNT0                = 0x6F @0x46
TCNT1                = 0x13 @0x84
TCNT1H               = 0x00 @0x85
TCNT1L               = 0x86 @0x84
TCNT2                = 0x59 @0xB2
TIFR0                = 0x07 @0x35
TIFR1                = 0x27 @0x36
TIFR2                = 0x07 @0x37
TIMSK0               = 0x00 @0x6E
TIMSK1               = 0x00 @0x6F
TIMSK2               = 0x00 @0x70
TWAMR                = 0x00 @0xBD
TWAR                 = 0xFE @0xBA
TWBR                 = 0x00 @0xB8
TWCR                 = 0x00 @0xBC
TWDR                 = 0xFF @0xBB
TWSR                 = 0xF8 @0xB9
UBRR0                = 0x00 @0xC4
UBRR0H               = 0x00 @0xC5
UBRR0L               = 0x00 @0xC4
UCSR0A               = 0x20 @0xC0
UCSR0B               = 0x00 @0xC1
```

```
UCSR0C                  = 0x06 @0xC2
UDR0                    = 0x00 @0xC6
WDTCSR                  = 0x0E @0x60
```

CHAPTER

SEVEN

HARDWARE

http://vusb.wikidot.com/hardware

I use Solution B:

“Solution B: Level conversion on D+ and D- Level conversion with Zener diodes.

Instead of reducing the AVR’s power supply, we can limit the output voltage on D+ and D- with Zener diodes. We recommend 3.6 V low power types, those that look like 1N4148 (usually 500 mW or less). Low power types are required because they have less capacitance and thus cause less distortion on the data lines. And 3.6 V is better than 3.3 V because 3.3 V diodes yield only ca. 2.7 V in conjunction with an 1.5 kâ„¦ (or more exactly 10 kâ„¦) pull-up resistor. With 3.3 V diodes, the device may not be detected reliably.

If you use Zener diodes for level conversion, please measure the voltage levels to make sure that the diodes you have chosen match the requirements.

Advantages of the Zener diode approach:

- Low cost.
- Easy to obtain.
- Entire design can be at 5 V.
- AVR can be clocked at high rates.

Disadvantages:

- Not a clean solution, a compromise between all parameters must be found.
- Zener diodes come with a broad range of characteristics, especially at low currents, results may not be reproducible.
- High currents when sending high-level.
- High level is different for signaling and in idle state because signaling uses high currents to drive the diodes while idle state is driven by a 1.5 kâ„¦ pull-up resistor.”

7.1 Pins

USB pins are defined in `pinconfig.h`:

```
#define USB_CFG_IOPORTNAME      D
/* This is the port where the USB bus is connected. When you configure it to
 * "B", the registers PORTB, PINB and DDRB will be used.
 */
#define USB_CFG_DMINUS_BIT      4
/* This is the bit number in USB_CFG_IOPORT where the USB D- line is connected.
 * This may be any bit in the port.
 */
#define USB_CFG_DPLUS_BIT       2
```

```
/* This is the bit number in USB_CFG_IOPORT where the USB D+ line is connected.
 * This may be any bit in the port. Please note that D+ must also be connected
 * to interrupt pin INT0! [You can also use other interrupts, see section
 * "Optional MCU Description" below, or you can connect D- to the interrupt, as
 * it is required if you use the USB_COUNT_SOF feature. If you use D- for the
 * interrupt, the USB interrupt will also be triggered at Start-Of-Frame
 * markers every millisecond.]
 */
```

Pin mapping depends on board. Example:

http://arduino.cc/hu/Hacking/PinMapping

CHAPTER

EIGHT

BUILD TESTS

Results:

index	board	Simple
1	atmega8	OK (P:3338 D:143)
2	atmega88	OK (P:3646 D:143)
3	bt	OK (P:3790 D:145)
4	bt328	OK (P:3786 D:145)
5	diecimila	OK (P:3790 D:145)
6	fio	ERR
7	lilypad	ERR
8	lilypad328	ERR
9	mega	OK (P:5036 D:145)
10	mega2560	OK (P:5040 D:145)
11	metaboard	OK (P:3790 D:145)
12	mini	OK (P:3790 D:145)
13	pro	ERR
14	pro328	ERR
15	pro5v	OK (P:3790 D:145)
16	pro5v328	OK (P:3786 D:145)
17	uno	OK (P:3786 D:145)
18	arduino_OrangutanSVP1284	OK (P:4120 D:147)
19	arduino_amber128	ERR
20	arduino_android2561	ERR
21	arduino_android2561_16	OK (P:4806 D:145)
22	arduino_at90can128	OK (P:4336 D:145)
23	arduino_at90can32	OK (P:4326 D:145)
24	arduino_at90can64	OK (P:4326 D:145)
25	arduino_at90usb162	OK (P:3730 D:145)
26	arduino_at90usb646	OK (P:4354 D:145)
27	arduino_at90usb647	OK (P:4458 D:145)
28	arduino_at90usbkey	OK (P:4474 D:147)
29	arduino_atmega16	ERR
30	arduino_atmega165	ERR
31	arduino_atmega3290p	OK (P:4054 D:147)
32	arduino_atmega8515	OK (P:3340 D:145)
33	arduino_atmega8535	OK (P:3452 D:145)
34	arduino_attiny2313	ERR
35	arduino_attiny26	ERR
36	arduino_attiny45	ERR
37	arduino_attiny85	ERR
38	arduino_bahbots1284p	ERR
39	arduino_butterfly	ERR

Continued on next page

Table 8.1 – continued from previous page

index	board	Simple
40	arduino_cerebot_plus	ERR
41	arduino_cerebotii	ERR
42	arduino_digilent_explorer	ERR
43	arduino_duino644	OK (P:3856 D:145)
44	arduino_duino644p	OK (P:3896 D:145)
45	arduino_gator	OK (P:3864 D:145)
46	arduino_illuminato	OK (P:3798 D:145)
47	arduino_penguino_avr	OK (P:3522 D:143)
48	arduino_teensy2_ser	OK (P:4604 D:145)
49	arduino_teensypp2_ser	OK (P:4370 D:147)
50	arduino_wiring1281	OK (P:4816 D:145)
51	atmega168	OK (P:3788 D:145)
52	atmega328	OK (P:3784 D:145)
53	atmega48	OK (P:3728 D:143)
54	atmega640	OK (P:5078 D:145)
55	atmega8	OK (P:3336 D:143)
56	atmega88	OK (P:3726 D:145)
57	bt	OK (P:3788 D:145)
58	bt328	OK (P:3784 D:145)
59	diecimila	OK (P:3788 D:145)
60	dvk90can1	ERR
61	ecavr_atmega32	ERR
62	fio	ERR
63	lilypad	ERR
64	lilypad328	ERR
65	mega	OK (P:5034 D:145)
66	mega1280stk500v2	OK (P:5034 D:145)
67	mega2560stk500v2	OK (P:5038 D:145)
68	mini	OK (P:3788 D:145)
69	pro	ERR
70	pro328	ERR
71	pro5v	OK (P:3788 D:145)
72	pro5v328	OK (P:3784 D:145)
73	stk502	ERR
74	stk525	ERR
75	stk525_647	ERR

Board configuration:

index	package	id	name	MCU
1	arduino	atmega8	Arduino NG or older w/ ATmega8	atmega8
2	arduino	atmega88	atmega88@20000000 programmer:usbasp	atmega8
3	arduino	bt	Arduino BT w/ ATmega168	atmega1
4	arduino	bt328	Arduino BT w/ ATmega328	atmega3
5	arduino	diecimila	Arduino Diecimila, Duemilanove, or Nano w/ ATmega168	atmega1
6	arduino	fio	Arduino Fio	atmega3
7	arduino	lilypad	LilyPad Arduino w/ ATmega168	atmega1
8	arduino	lilypad328	LilyPad Arduino w/ ATmega328	atmega3
9	arduino	mega	Arduino Mega (ATmega1280)	atmega1
10	arduino	mega2560	Arduino Mega 2560	atmega2
11	arduino	metaboard	Metaboard	atmega1
12	arduino	mini	Arduino Mini	atmega1
13	arduino	pro	Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168	atmega1

Con

Table 8.2 – continued from previous page

index	package	id	name	MCU
14	arduino	pro328	Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328	atmega3
15	arduino	pro5v	Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168	atmega1
16	arduino	pro5v328	Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328	atmega3
17	arduino	uno	Arduino Uno	atmega3
18	arduino-extras	arduino_OrangutanSVP1284	Arduino-Orangutan SVP-1284	atmega1
19	arduino-extras	arduino_amber128	Arduino-Amber 128 14.7456 Mhz	atmega1
20	arduino-extras	arduino_android2561	Arduino-Android 2561 8Mhz	atmega2
21	arduino-extras	arduino_android2561_16	Arduino-Android 2561 16Mhz	atmega2
22	arduino-extras	arduino_at90can128	AT90CAN128 development board NHL (arduino core)	at90can
23	arduino-extras	arduino_at90can32	at90can32 (arduino core)	at90can3
24	arduino-extras	arduino_at90can64	at90can64 (arduino core)	at90can6
25	arduino-extras	arduino_at90usb162	Arduino-at90usb162	at90usb
26	arduino-extras	arduino_at90usb646	Arduino-at90usb646	at90usb6
27	arduino-extras	arduino_at90usb647	Arduino-at90usb647	at90usb6
28	arduino-extras	arduino_at90usbkey	Arduino-at90usbkey	at90usb
29	arduino-extras	arduino_atmega16	Arduino-Atmega16	atmega1
30	arduino-extras	arduino_atmega165	Arduino-Atmega165	atmega1
31	arduino-extras	arduino_atmega3290p	Arduino-Atmega3290p	atmega3
32	arduino-extras	arduino_atmega8515	Arduino-ATmega8515	atmega8
33	arduino-extras	arduino_atmega8535	Arduino-Test-Atmega8535	atmega8
34	arduino-extras	arduino_attiny2313	Arduino-ATtiny2313	attiny23
35	arduino-extras	arduino_attiny26	Arduino-ATtiny26	attiny26
36	arduino-extras	arduino_attiny45	Arduino-ATtiny45	attiny45
37	arduino-extras	arduino_attiny85	Arduino-ATtiny85	attiny85
38	arduino-extras	arduino_bahbots1284p	Arduino-BahBots 1284p	atmega1
39	arduino-extras	arduino_butterfly	Arduino-Butterfly stk500	atmega1
40	arduino-extras	arduino_cerebot_plus	Arduino-Cerebot Plus	atmega2
41	arduino-extras	arduino_cerebotii	Arduino-Cerebot II atemga64	atmega6
42	arduino-extras	arduino_digilent_explorer	Arduino-Digilent I/O Explorer USB	atmega1
43	arduino-extras	arduino_duino644	Arduino-Duino 644	atmega6
44	arduino-extras	arduino_duino644p	Arduino-Duino 644P	atmega6
45	arduino-extras	arduino_gator	Arduino-Rugged Circuits Gator Board	atmega3
46	arduino-extras	arduino_illuminato	Arduino-illuminato	atmega6
47	arduino-extras	arduino_penguino_avr	Arduino-Penguino AVR	atmega3
48	arduino-extras	arduino_teensy2_ser	Arduino-Teensy 2.0 (USB Serial)	atmega3
49	arduino-extras	arduino_teensypp2_ser	Arduino-Teensy++ 2.0 (USB Serial)	at90usb
50	arduino-extras	arduino_wiring1281	Arduino-Wiring 1281	atmega1
51	arduino-extras	atmega168	Arduino NG or older w/ ATmega168	atmega1
52	arduino-extras	atmega328	Arduino Duemilanove or Nano w/ ATmega328	atmega3
53	arduino-extras	atmega48	Arduino Atmega48	atmega4
54	arduino-extras	atmega640	Arduino atmega640	atmega6
55	arduino-extras	atmega8	Arduino NG or older w/ ATmega8	atmega8
56	arduino-extras	atmega88	Atmega88	atmega8
57	arduino-extras	bt	Arduino BT w/ ATmega168	atmega1
58	arduino-extras	bt328	Arduino BT w/ ATmega328	atmega3
59	arduino-extras	diecimila	Arduino Diecimila, Duemilanove, or Nano w/ ATmega168	atmega1
60	arduino-extras	dvk90can1	STK500 w/DVK90CAN1 - AT90can128 (Arduino Core)	at90can
61	arduino-extras	ecavr_atmega32	Embedded market atmega32	atmega3
62	arduino-extras	fio	Arduino Fio	atmega3
63	arduino-extras	lilypad	LilyPad Arduino w/ ATmega168	atmega1
64	arduino-extras	lilypad328	LilyPad Arduino w/ ATmega328	atmega3
65	arduino-extras	mega	Arduino Mega	atmega1
66	arduino-extras	mega1280stk500v2	Arduino Mega1280 stk500v2	atmega1

Con

Table 8.2 – continued from previous page

index	package	id	name	MCU
67	arduino-extras	mega2560stk500v2	Arduino Mega2560 stk500v2	atmega2
68	arduino-extras	mini	Arduino Mini	atmega1
69	arduino-extras	pro	Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168	atmega1
70	arduino-extras	pro328	Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328	atmega3
71	arduino-extras	pro5v	Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168	atmega1
72	arduino-extras	pro5v328	Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328	atmega3
73	arduino-extras	stk502	STK500 w/STKk502 - ATmega169 (Arduino Core)	atmega1
74	arduino-extras	stk525	STK500 w/STK525 - at90usb1287 (Arduino Core)	at90usb
75	arduino-extras	stk525_647	STK500 w/STK525 - at90usb647 (Arduino Core)	at90usb

CHAPTER

NINE

DOXYGEN DOCUMENTATION

Files