Water Tank Level Monitor

Post a reply

14 posts

by **mikey11** » Wed Jun 26, 2013 4:23 am

Greets,

I built my first project, and it's complete enough to throw out to the masses. I live in an acreage community where we have to have our water trucked in. My water guy says he always brings in 5m3 of water and fills my tank until it starts to overflow onto the ground. I asked him if he would use an electronic level monitor to offload, and he seemed keen on the idea, so I had my impetus to get started. He also expressed some concern that my tank may have a leak in a portion of the tank. I figured that I could monitor the tank better this way and determine if this is true, and the height where the leak is at.

I bought the following equipment:

A model B rpi: $40
http://www.adafruit.com/products/998

A 16X2 rgb pi plate controlled by I2C: $25
http://www.adafruit.com/products/1110

Image

Posts: 349

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

I recycled a 5V power supply, and had an SD card sitting around as well. I would estimate those to cost $25.

I also used a half beer can, and recycled some foam that was used in packaging.

Total cost: ~$125 USD

I installed Occidentalis v0.2 on the SD card as I heard this distribution was primed for using the GPIO. I downloaded the example code to run the pi-plate from adafruit, and then adapted the loop they use to detect button presses to handle my level monitoring code which uses the LCD code to output level monitoring data to the pi plate.

I would like to spend a few minutes on the ultrasonic sensor.

http://www.maxbotix.com/documents/HRLV-MaxSonar-EZ_Datasheet.pdf

I used the sensor in serial mode, and attached vcc to the 3.3v pin on the GPIO header, with gnd to a gnd on the header. I used pin 5 on the sensor (tx) and tied it the serial rx on the rpi GPIO header. The sensor can use 3.3v or 5v, but I opted for 3.3v, because I read somewhere that the rpi's GPIO pins do not like anything above 3.3v as input. The serial output from the sensor is thus kept at safe voltage.

It has been a long time since I've used serial input, and I was completely new to python, and just wanted things to 'work' without getting too complicated. The sensor put out in rs232, but python did not read this as anything but garbage, so I soldered the jumper on the back of the sensor. This converted the output to TTL, and everything worked fine. I guess the rpi likes TTL input for serial.

The sensor puts out "RXXXX<CR>" where XXXX is a numeric value between 0300 (min distance) and 5000 (max distance) these are measured in millimeters. I take a ten character snapshot of the buffer, search for the R, then grab the numeric info following the R which I convert from a string to a number for calculations.

Before I could get serial input on the GPIO, I had to disable it from being used elsewhere in the standard bootup of linux. I followed the advice on this page for commenting out/removing references to the GPIO serial port (/dev/ttyAMA0) from the startup files.

The reason you have to do this is that the serial port is enabled to be used as a debugging

http://www.hobbytronics.co.uk/raspberry-pi-serial-port

I took measurements of my water tank and converted them to mm to use with the sensor programming.

Basically there is a deep/wide cylinder on the bottom to store water, with a narrow cylinder at the top. I want to fill only to the bottom of the narrow cylinder, as with these tanks, it is highly likely that it will leak at this junction anyways, thereby wasting water. I have to account for this offset in my programming.

This is sort of what it would look like, but the bottom tank diameter is wider than this pic:

I found my tank to be 3733mm deep (3.7 meters), with the smaller cylinder having a depth of 787mm (just under a meter).

This gives my tank a useable depth of ~3000 mm. The values fall within the sensor's abilities of 300mm-5000mm, so at least for the range, the sensor was the right pick.

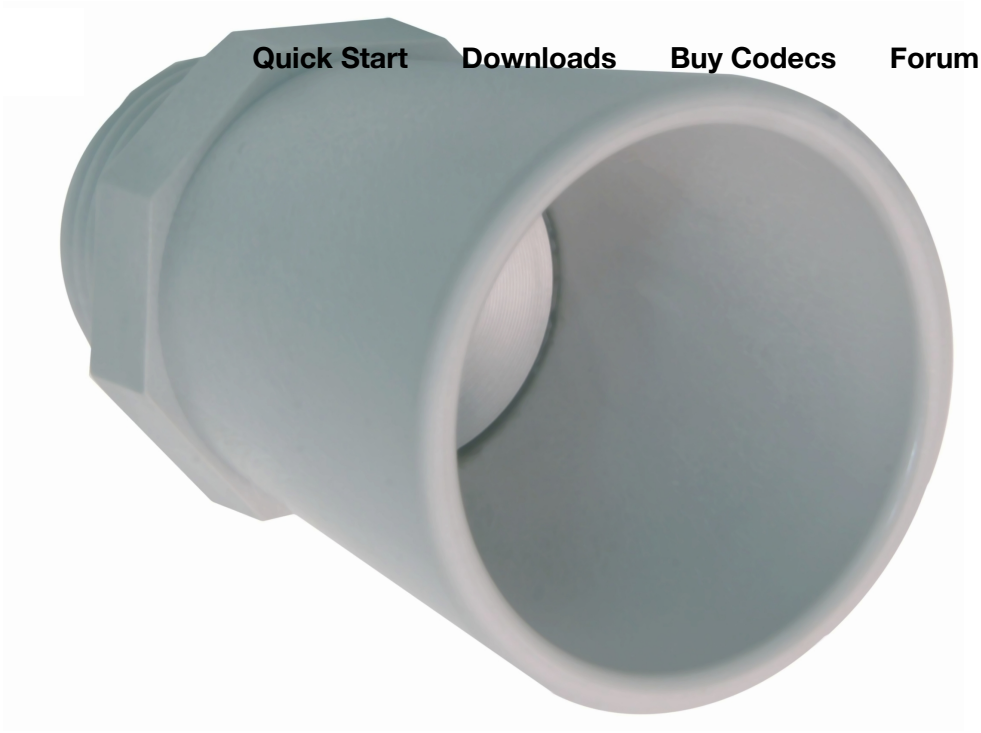
I don't know my tanks diameter yet, but I will just wait for my next delivery and use geometry to calculate based off the volume that gets offloaded, or wait for next years spring cleaning. With the tank diameter, I can calculate the water volume in the tank, which I can't do now. At the current time though, I can calculate % full of the useable space.

I eventually determined that the sensor was picking up the walls of the narrow cylinder at the top, and was unable to read the bottom. I saw this page:

http://www.robot-electronics.co.uk/htm/sonar_faq.htm

Then I cut a beer can in half, mounted the sensor in the bottom, and covered the walls with foam. This worked, and I could accurately sense the water level. I double checked the readings I was getting with a tape measure. Everything checked out.

For your reference, here is an example of a purpose built sensor for the task, notice the price.



http://www.maxbotix.com/Ultrasonic_Sensors/MB7040.htm

See my short video:

http://www.youtube.com/watch?v=aPnvxh_UWuc

I made my program which went pretty good, and this is my latest version. Almost all my new code happens at the bottom. I added the import for serial and sleep at the top, and put my measured values for the tank there.

Code: Select all

```
#!/usr/bin/python

# Python library for Adafruit RGB-backlit LCD plate for Raspberry
Pi.
# Written by Adafruit Industries.  MIT license.

# This is essentially a complete rewrite, but the calling syntax
# and constants are based on code from lrvick and LiquidCrystal.
# lrvic - https://github.com/lrvick/raspi-hd44780/blob/master
/hd44780.py
# LiquidCrystal - https://github.com/arduino/Arduino/blob/master
/libraries/LiquidCrystal/LiquidCrystal.cpp
```

```
from time import sleep
import serial

#The lower distance value is determind experimentally on-site with
the tank empty
#Upper distance value is the height you want to fill to.
#these work together to make the progress barad % full
upperdistance = 787 #distance from sensor to full tank
lowerdistance = 3733 # The distance to tank bottom


class Adafruit_CharLCDPlate(Adafruit_I2C):

    #
------------------------------------------------------------------------
    # Constants

    # Port expander registers
    MCP23017_IOCON_BANK0    = 0x0A  # IOCON when Bank 0 active
    MCP23017_IOCON_BANK1    = 0x15  # IOCON when Bank 1 active
    # These are register addresses when in Bank 1 only:
    MCP23017_GPIOA          = 0x09
    MCP23017_IODIRB         = 0x10
    MCP23017_GPIOB          = 0x19

    # Port expander input pin definitions
    SELECT                  = 0
    RIGHT                   = 1
    DOWN                    = 2
    UP                      = 3
    LEFT                    = 4

    # LED colors
    OFF                     = 0x00
    RED                     = 0x01
    GREEN                   = 0x02
    BLUE                    = 0x04
    YELLOW                  = RED + GREEN
```

```
    WHITE                   = RED + GREEN + BLUE
    ON                      = RED + GREEN + BLUE

    # LCD Commands
    LCD_CLEARDISPLAY        = 0x01
    LCD_RETURNHOME          = 0x02
    LCD_ENTRYMODESET        = 0x04
    LCD_DISPLAYCONTROL      = 0x08
    LCD_CURSORSHIFT         = 0x10
    LCD_FUNCTIONSET         = 0x20
    LCD_SETCGRAMADDR        = 0x40
    LCD_SETDDRAMADDR        = 0x80

    # Flags for display on/off control
    LCD_DISPLAYON           = 0x04
    LCD_DISPLAYOFF          = 0x00
    LCD_CURSORON            = 0x02
    LCD_CURSOROFF           = 0x00
    LCD_BLINKON             = 0x01
    LCD_BLINKOFF            = 0x00

    # Flags for display entry mode
    LCD_ENTRYRIGHT          = 0x00
    LCD_ENTRYLEFT           = 0x02
    LCD_ENTRYSHIFTINCREMENT = 0x01
    LCD_ENTRYSHIFTDECREMENT = 0x00

    # Flags for display/cursor shift
    LCD_DISPLAYMOVE = 0x08
    LCD_CURSORMOVE  = 0x00
    LCD_MOVERIGHT   = 0x04
    LCD_MOVELEFT    = 0x00

    #
----------------------------------------------------------------------
    # Constructor
```

```
        self.i2c = Adafruit_I2C(addr, busnum, debug)

        # I2C is relatively slow.  MCP output port states are cached
        # so we don't need to constantly poll-and-change bit states.
        self.porta, self.portb, self.ddrb = 0, 0, 0b00010000

        # Set MCP23017 IOCON register to Bank 0 with sequential
operation.
        # If chip is already set for Bank 0, this will just write to
OLATB,
        # which won't seriously bother anything on the plate right
now
        # (blue backlight LED will come on, but that's done in the
next
        # step anyway).
        self.i2c.bus.write_byte_data(
          self.i2c.address, self.MCP23017_IOCON_BANK1, 0)

        # Brute force reload ALL registers to known state.  This
also
        # sets up all the input pins, pull-ups, etc. for the Pi
Plate.
        self.i2c.bus.write_i2c_block_data(
          self.i2c.address, 0,
          [ 0b00111111,   # IODIRA    R+G LEDs=outputs,
buttons=inputs
            self.ddrb ,   # IODIRB    LCD D7=input, Blue LED=output
            0b00111111,   # IPOLA     Invert polarity on button
inputs
            0b00000000,   # IPOLB
            0b00000000,   # GPINTENA  Disable interrupt-on-change
            0b00000000,   # GPINTENB
            0b00000000,   # DEFVALA
            0b00000000,   # DEFVALB
            0b00000000,   # INTCONA
            0b00000000,   # INTCONB
            0b00000000,   # IOCON
            0b00000000,   # IOCON
```

```
            0b00000000,   # INTFA
            0b00000000,   # INTFB
            0b00000000,   # INTCAPA
            0b00000000,   # INTCAPB
            self.porta,   # GPIOA
            self.portb,   # GPIOB
            self.porta,   # OLATA     0 on all outputs; side effect
of
            self.portb ]) # OLATB     turning on R+G+B backlight
LEDs.

        # Switch to Bank 1 and disable sequential operation.
        # From this point forward, the register addresses do NOT
match
        # the list immediately above.  Instead, use the constants
defined
        # at the start of the class.  Also, the address register
will no
        # longer increment automatically after this -- multi-byte
        # operations must be broken down into single-byte calls.
        self.i2c.bus.write_byte_data(
          self.i2c.address, self.MCP23017_IOCON_BANK0, 0b10100000)

        self.displayshift   = (self.LCD_CURSORMOVE |
                               self.LCD_MOVERIGHT)
        self.displaymode    = (self.LCD_ENTRYLEFT |
                               self.LCD_ENTRYSHIFTDECREMENT)
        self.displaycontrol = (self.LCD_DISPLAYON |
                               self.LCD_CURSOROFF |
                               self.LCD_BLINKOFF)

        self.write(0x33) # Init
        self.write(0x32) # Init
        self.write(0x28) # 2 line 5x8 matrix
        self.write(self.LCD_CLEARDISPLAY)
        self.write(self.LCD_CURSORSHIFT    | self.displayshift)
        self.write(self.LCD_ENTRYMODESET   | self.displaymode)
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)
```

```
    #
--------------------------------------------------------------------
    # Write operations

    # The LCD data pins (D4-D7) connect to MCP pins 12-9 (PORTB4-1),
in
    # that order.  Because this sequence is 'reversed,' a direct
shift
    # won't work.  This table remaps 4-bit data values to MCP PORTB
    # outputs, incorporating both the reverse and shift.
    flip = ( 0b00000000, 0b00010000, 0b00001000, 0b00011000,
             0b00000100, 0b00010100, 0b00001100, 0b00011100,
             0b00000010, 0b00010010, 0b00001010, 0b00011010,
             0b00000110, 0b00010110, 0b00001110, 0b00011110 )

    # Low-level 4-bit interface for LCD output.  This doesn't
actually
    # write data, just returns a byte array of the PORTB state over
time.
    # Can concatenate the output of multiple calls (up to 8) for
more
    # efficient batch write.
    def out4(self, bitmask, value):
        hi = bitmask | self.flip[value >> 4]
        lo = bitmask | self.flip[value & 0x0F]
        return [hi | 0b00100000, hi, lo | 0b00100000, lo]


    # The speed of LCD accesses is inherently limited by I2C through
the
    # port expander.  A 'well behaved program' is expected to poll
the
    # LCD to know that a prior instruction completed.  But the
timing of
    # most instructions is a known uniform 37 mS.  The enable strobe
    # can't even be twiddled that fast through I2C, so it's a safe
bet
```

```
    # several I2C transfers for reconfiguring the port direction).
    # The D7 pin is set as input when a potentially time-consuming
    # instruction has been issued (e.g. screen clear), as well as on
    # startup, and polling will then occur before more commands or
data
    # are issued.

    pollables = ( LCD_CLEARDISPLAY, LCD_RETURNHOME )

    # Write byte, list or string value to LCD
    def write(self, value, char_mode=False):
        """ Send command/data to LCD """

        # If pin D7 is in input state, poll LCD busy flag until
clear.
        if self.ddrb & 0b00010000:
            lo = (self.portb & 0b00000001) | 0b01000000
            hi = lo | 0b00100000 # E=1 (strobe)
            self.i2c.bus.write_byte_data(
              self.i2c.address, self.MCP23017_GPIOB, lo)
            while True:
                # Strobe high (enable)
                self.i2c.bus.write_byte(self.i2c.address, hi)
                # First nybble contains busy state
                bits = self.i2c.bus.read_byte(self.i2c.address)
                # Strobe low, high, low.  Second nybble (A3) is
ignored.
                self.i2c.bus.write_i2c_block_data(
                  self.i2c.address, self.MCP23017_GPIOB, [lo, hi,
lo])
                if (bits & 0b00000010) == 0: break # D7=0, not busy
            self.portb = lo

            # Polling complete, change D7 pin to output
            self.ddrb &= 0b11101111
            self.i2c.bus.write_byte_data(self.i2c.address,
              self.MCP23017_IODIRB, self.ddrb)
```

```
        if char_mode: bitmask |= 0b10000000 # Set data bit if not a
command

        # If string or list, iterate through multiple write ops
        if isinstance(value, str):
            last = len(value) - 1 # Last character in string
            data = []             # Start with blank list
            for i, v in enumerate(value): # For each character...
                # Append 4 bytes to list representing PORTB over
time.
                # First the high 4 data bits with strobe (enable)
set
                # and unset, then same with low 4 data bits (strobe
1/0).
                data.extend(self.out4(bitmask, ord(v)))
                # I2C block data write is limited to 32 bytes max.
                # If limit reached, write data so far and clear.
                # Also do this on last byte if not otherwise
handled.
                if (len(data) >= 32) or (i == last):
                    self.i2c.bus.write_i2c_block_data(
                      self.i2c.address, self.MCP23017_GPIOB, data)
                    self.portb = data[-1] # Save state of last byte
out
                    data       = []       # Clear list for next
iteration
        elif isinstance(value, list):
            # Same as above, but for list instead of string
            last = len(value) - 1
            data = []
            for i, v in enumerate(value):
                data.extend(self.out4(bitmask, v))
                if (len(data) >= 32) or (i == last):
                    self.i2c.bus.write_i2c_block_data(
                      self.i2c.address, self.MCP23017_GPIOB, data)
                    self.portb = data[-1]
                    data       = []
        else:
```

```
            self.i2c.bus.write_i2c_block_data(
              self.i2c.address, self.MCP23017_GPIOB, data)
            self.portb = data[-1]

        # If a poll-worthy instruction was issued, reconfigure D7
        # pin as input to indicate need for polling on next call.
        if (not char_mode) and (value in self.pollables):
            self.ddrb |= 0b00010000
            self.i2c.bus.write_byte_data(self.i2c.address,
              self.MCP23017_IODIRB, self.ddrb)


    #
-----------------------------------------------------------------------------
    # Utility methods

    def begin(self, cols, lines):
        self.currline = 0
        self.numlines = lines
        self.clear()


    # Puts the MCP23017 back in Bank 0 + sequential write mode so
    # that other code using the 'classic' library can still work.
    # Any code using this newer version of the library should
    # consider adding an atexit() handler that calls this.
    def stop(self):
        self.porta = 0b11000000  # Turn off LEDs on the way out
        self.portb = 0b00000001
        sleep(0.0015)
        self.i2c.bus.write_byte_data(
          self.i2c.address, self.MCP23017_IOCON_BANK1, 0)
        self.i2c.bus.write_i2c_block_data(
          self.i2c.address, 0,
          [ 0b00111111,   # IODIRA
            self.ddrb ,   # IODIRB
            0b00000000,   # IPOLA
            0b00000000,   # IPOLB
```

```
            0b00000000,   # DEFVALA
            0b00000000,   # DEFVALB
            0b00000000,   # INTCONA
            0b00000000,   # INTCONB
            0b00000000,   # IOCON
            0b00000000,   # IOCON
            0b00111111,   # GPPUA
            0b00000000,   # GPPUB
            0b00000000,   # INTFA
            0b00000000,   # INTFB
            0b00000000,   # INTCAPA
            0b00000000,   # INTCAPB
            self.porta,   # GPIOA
            self.portb,   # GPIOB
            self.porta,   # OLATA
            self.portb ]) # OLATB


    def clear(self):
        self.write(self.LCD_CLEARDISPLAY)


    def home(self):
        self.write(self.LCD_RETURNHOME)


    row_offsets = ( 0x00, 0x40, 0x14, 0x54 )
    def setCursor(self, col, row):
        if row > self.numlines: row = self.numlines - 1
        elif row < 0:           row = 0
        self.write(self.LCD_SETDDRAMADDR | (col +
self.row_offsets[row]))


    def display(self):
        """ Turn the display on (quickly) """
        self.displaycontrol |= self.LCD_DISPLAYON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)
```

```
    def noDisplay(self):
        """ Turn the display off (quickly) """
        self.displaycontrol &= ~self.LCD_DISPLAYON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def cursor(self):
        """ Underline cursor on """
        self.displaycontrol |= self.LCD_CURSORON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def noCursor(self):
        """ Underline cursor off """
        self.displaycontrol &= ~self.LCD_CURSORON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def ToggleCursor(self):
        """ Toggles the underline cursor On/Off """
        self.displaycontrol ^= self.LCD_CURSORON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def blink(self):
        """ Turn on the blinking cursor """
        self.displaycontrol |= self.LCD_BLINKON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def noBlink(self):
        """ Turn off the blinking cursor """
        self.displaycontrol &= ~self.LCD_BLINKON
        self.write(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def ToggleBlink(self):
        """ Toggles the blinking cursor """
```

```
    def scrollDisplayLeft(self):
        """ These commands scroll the display without changing the
RAM """
        self.displayshift = self.LCD_DISPLAYMOVE | self.LCD_MOVELEFT
        self.write(self.LCD_CURSORSHIFT | self.displayshift)

    def scrollDisplayRight(self):
        """ These commands scroll the display without changing the
RAM """
        self.displayshift = self.LCD_DISPLAYMOVE |
self.LCD_MOVERIGHT
        self.write(self.LCD_CURSORSHIFT | self.displayshift)

    def leftToRight(self):
        """ This is for text that flows left to right """
        self.displaymode |= self.LCD_ENTRYLEFT
        self.write(self.LCD_ENTRYMODESET | self.displaymode)

    def rightToLeft(self):
        """ This is for text that flows right to left """
        self.displaymode &= ~self.LCD_ENTRYLEFT
        self.write(self.LCD_ENTRYMODESET | self.displaymode)

    def autoscroll(self):
        """ This will 'right justify' text from the cursor """
        self.displaymode |= self.LCD_ENTRYSHIFTINCREMENT
        self.write(self.LCD_ENTRYMODESET | self.displaymode)

    def noAutoscroll(self):
        """ This will 'left justify' text from the cursor """
        self.displaymode &= ~self.LCD_ENTRYSHIFTINCREMENT
```

```
    def createChar(self, location, bitmap):
        self.write(self.LCD_SETCGRAMADDR | ((location & 7) << 3))
        self.write(bitmap, True)
        self.write(self.LCD_SETDDRAMADDR)


    def message(self, text):
        """ Send string to LCD. Newline wraps to second line"""
        lines = str(text).split('\n')    # Split at newline(s)
        for i, line in enumerate(lines): # For each substring...
            if i > 0:                    # If newline(s),
                self.write(0xC0)         #  set DDRAM address to 2nd
line
            self.write(line, True)       # Issue substring


    def backlight(self, color):
        c          = ~color
        self.porta = (self.porta & 0b00111111) | ((c & 0b011) << 6)
        self.portb = (self.portb & 0b11111110) | ((c & 0b100) >> 2)
        # Has to be done as two writes because sequential operation
is off.
        self.i2c.bus.write_byte_data(
          self.i2c.address, self.MCP23017_GPIOA, self.porta)
        self.i2c.bus.write_byte_data(
          self.i2c.address, self.MCP23017_GPIOB, self.portb)


    # Read state of single button
    def buttonPressed(self, b):
        return (self.i2c.readU8(self.MCP23017_GPIOA) >> b) & 1


    # Read and return bitmask of combined button state
    def buttons(self):
        return self.i2c.readU8(self.MCP23017_GPIOA) & 0b11111
```

```
------------------------------------------------------------------------
    # Test code
maxbotix = serial.Serial("/dev
/ttyAMA0",baudrate=9600,timeout=5)#other examples that didnt specify
baudrate and timeout did not work for me


if __name__ == '__main__':

    lcd = Adafruit_CharLCDPlate()
    lcd.begin(16, 2)
    lcd.clear()
    lcd.message("Anzac Cistern \nLevel Monitor")
    sleep(3)

    col = (('Red' , lcd.RED) , ('Yellow', lcd.YELLOW), ('Green' ,
lcd.GREEN),
           ('Teal', lcd.TEAL), ('Blue'  , lcd.BLUE)  , ('Violet',
lcd.VIOLET),
           ('Off' , lcd.OFF) , ('On'    , lcd.ON))

    print "Cycle thru backlight colors"
    for c in col:
       print c[0]
       #lcd.clear()
       #lcd.message(c[0])
       lcd.backlight(c[1])
       sleep(0.5)

    btn = ((lcd.SELECT, 'Select', lcd.ON),
           (lcd.LEFT  , 'Left'  , lcd.RED),
           (lcd.UP    , 'Up'    , lcd.BLUE),
           (lcd.DOWN  , 'Down'  , lcd.GREEN),
           (lcd.RIGHT , 'Right' , lcd.VIOLET))
    b=9

    while True:
        for b in btn:
```

```
if you don't do this, you won't get responsive readings
            currdistance = maxbotix.readline(10) #Take ten
characters worth of the serial buffer than accumulates since the
flush
            stripstart = currdistance.find("R") #Look for the
character "R", find out where it occurs in the string first
            stripend = stripstart + 5 #Define the end of the
character length we need to grab the numeric info
            currdistance = currdistance[stripstart+1:stripend]
#strip out the numeric info
            #print currdistance
            currmm = float(currdistance) #Now make the info a number
instead of a string
            #print currmm
            percentfull = (lowerdistance-(currmm-
upperdistance))/lowerdistance*100 #calculate the percentage full
using the actual tank measurements
            if percentfull <0:
                lcd.backlight(5) #Show that the level measured is
out of useful range Indicates that you need to measure total depth
of tank again
            if percentfull >0 and percentfull <50:
                lcd.backlight(3)
            if percentfull >50 and percentfull <80:
                lcd.backlight(2)
            if percentfull >80 and percentfull <100.1:
                lcd.backlight(1) #Red backlight for a tank that is
almost full
            if percentfull > 100.1:
                lcd.backlight (6) #Indicate overfull situation this
would be a good place to attach an audible alarm like a buzzer fired
by a relay for 2 seconds
            #print percentfull
            progressbar = ((percentfull/100)*16) #Its a 16X2
display. How many spaces of the first line need to be lit up?
            #print progressbar
            progressbarstring = "*" * int(progressbar) #Light up the
first line with as many *'s as required to roughly correspond to the
```

```
a string so I can concatenate it with other string info
            percentfull = percentfull[0:4] #dont need too much
detail on the display cut down the # of decimal places from the left
            lcd.clear()
            lcd.message(progressbarstring + "\nmm: " + currdistance
+ " " + percentfull +"%") #Go to the second line, show distance in
mm, and % full
            wait = (1)

            if lcd.buttonPressed(b[0]):
                exit()
```

you also need another file, "Adafruit_I2C.py" or "Adafruit_I2C.pyc"

With these you should get good results.

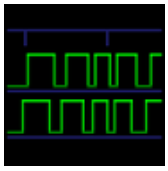
Ideas for improvement:

1. Calculate water volume in the tank, sms the water truck driver two days prior to delivery and give a volume required
2. data log the readings every minute for a spreadsheet
3. make it accessible over the internet
4. Add other I2C sensors like temp/anemometer

Last edited by mikey11 on Thu Jun 27, 2013 4:01 am, edited 2 times in total.

by **joan** » Wed Jun 26, 2013 7:46 am

Good job!



Posts: 12316

Joined: Thu Jul 05, 2012 5:09 pm

Location: UK

by **robbes** » Wed Jun 26, 2013 11:41 pm

Nicely done.

I have a question for you: I want to measure the depth of water in my well pipe, which is 6" diam. Do you think the sensor you used, with beer can modification, could provide a reasonably accurate measurement inside the pipe?

Posts: 78

Joined: Sun Jan 20, 2013 7:11 pm

Location: Canada - off the

by **mikey11** » Thu Jun 27, 2013 12:58 am
this type of sensor will probably not work, the beam spreads out too far for that. For really thin diameters and fluids, a float would probably be best.

I saw a cheap sensor idea for multiple water levels (not continuous sensing, but sensors placed at predetermined heights, and then you infer that the water level is between the heights. They just dangled wires with weights down, then when the water came to the level of the two wires, the change in conductivity was detected.

Posts: 349

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

by **MaxBotix Inc** » Wed Jul 10, 2013 9:30 pm
Hello, this is Tom Bonar from MaxBotix Inc.,

I saw your forum post and would like to lend some advice with the ultrasonic sensor.

Posts: 3

Joined: Wed Jul 10, 2013 9:29 pm

@Robbes

You state: I have a question for you: I want to measure the depth of water in my well pipe, which is 6" diam. Do you think the sensor you used, with beer can modification, could provide a reasonably accurate measurement inside the pipe?

Please note that when operating in a pipe with a diameter of 6" the ultrasonic sensor may not operate properly. Small diameter pipes are a challenging environment for the physics of ultrasonic sensors. In this environment it is recommended to test a sensor such as the MB7389, but the results are not guaranteed. It is recommended to use the largest diameter pipe possible.

@Mikey11

When using the Raspberry Pi to power any of the sensors, the 3.3V regulator must be used as the I/O connections on the RPi GPIO are not 5V tolerant. Being that our serial outputs are 0 to VCC level, when powering at 3.3V the serial output will be at 3.3V maximum.

The HRLV-MaxSonar-EZ sensor will work in your application for short term installations provided you protect the sensor from getting wet. If you are planning on using a sensor for long term installations, it is recommended to use the MB7389 because of its rugged design and IP67 rating.

When using a sensor such as an XL-MaxSonar or a MaxSonar-WR with a Raspberry Pi, It is highly recommended to use an external power supply. If you wish to power either of these sensor lines with a RaspberryPi, a power filter such as the MB7961 or better is required.

With this being said, it is not recommended to use the WR or XL series when directly powered by the Raspberry Pi, there may be unforeseen damage that can occur with extended use or varying conditions.

Our XL-MaxSonar and MaxSonar- WR sensors have up to a 100mA power draw that lasts 400uS. The regulator on the RaspberryPi is only rated for 50mA maximum. The power filter kit reduces the 100mA current spike that is drawn by the sensor.

Please let me know if you have any questions.
Best regards,

Tom Bonar
Technical Support
of MaxBotix Inc.
Phone: (218) 454-0766 ext 2#
Fax: (218) 454-0768
Email: thomas@maxbotix.com
Web: www.maxbotix.com

Follow us on Facebook at: http://www.facebook.com/pages/MaxBotix- ... 9384204938

Technical support and sales are subject to
the terms and conditions listed on our
website at http://www.maxbotix.com/

by **david4shure** » Mon Aug 12, 2013 6:56 pm

Posts: 1

Joined: Mon Aug 12, 2013 6:55 pm

Just out of curiousity, did you write the Adafruit_CharLCDPlate class or does that a pre written library?

by **txt3rob** » Tue Aug 13, 2013 9:07 am

Posts: 363

Joined: Sat Aug 11, 2012 3:45 pm

the the spreadsheet idea i would use google docs to write to.

sms there are many API's with python so thats not an issue.
The Raspberry Pi Hell Guy - Random Ramblings to assist me and others.
http://raspberrypihell.blogspot.com
My Github - http:/www.github.com/txt3rob/
http://www.smspi.co.uk - send free uk sms via your raspberry pi from here

by **MaxBotix Inc** » Thu Sep 05, 2013 9:35 pm

Posts: 3

Joined: Wed Jul 10, 2013 9:29 pm

Good Afternoon!
This is Tom Bonar from MaxBotix Inc.

We have recently published an article for connecting our sensor to a Raspberry Pi. We invite you to review this article. If you have any pictures of your setup that you would like to share with other Raspberry Pi users, please email them to us. We also welcome any code examples that you have, if you wish to share them with other users.

The MaxBotix Team, hopes you have a great day!

Tom Bonar
Technical Support and Webmaster
MaxBotix Inc
thomas@maxbotix.com

by **ripley119** » Mon Feb 10, 2014 6:26 pm

Posts: 1

Joined: Mon Feb 10, 2014

Bloody awesome! I'm not sure that I have the skills to really create this myself but I'm sure that I can find someone that can probably help me do it.

So this will be great for my rain tank, I'll be able to have the measurements in the house and

not 100% that it's going to have the ability or grunt to be able to handle it. Essentially I want to build my own fill level sensor for bottling, like beer & wine etc, so it would need to be able to measure density through a glass bottle and very quickly switch off the filler as it the liquid hits the right level - here's a link to something of industrial grade http://www.peco-inspx.com/fill-level-mo ... inspection but this thing uses gamma x-ray to be able to pass through the bottle.

So essentially my question is can the sensor that you've built here pass through glass and give me an accurate reading or is just not designed for this purpose?

Thanks!
by **mikey11** » Thu Feb 27, 2014 4:18 pm
I hadn't checked this post forever.

Posts: 349

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

I'm glad to see there was interest, and it's nice to see the maxbotix rep is keeping engaged with the community.

I used my setup without issue during last summer, but took it down for winter.

This year, I am going to winterize it, use a small 12V rechargeable lead acid battery with a solar panel to trickle charge it.

The pi is going to be set up to run for 25 minutes on a button press. This is good for the time it takes for the water guy to do his delivery. I may increase my code to have it take the initial and final water height and record it to a CSV for me to look at from time to time.

I saw one question about the libraries for the display. I took those from the adafruit website, and just replaced some of their example code with mine.

As for the concerns of the maxbotix rep, I'm going to have to take the time to re-read the message to understand it better. I wasn't too concerned, as it was a 3.3v device feeding info to the pi at 3.3v

I'm not power genius, so it's very likely I've overlooked something important.

Anyways, I just wanted to say that using the maxbotix sensor was VERY easy, and it just worked exactly as advertised right away. I would definitely purchase more sensors from them as time goes on.

edit: I re-read the post from the maxbotix rep. Indeed the current issue is a problem. I have never fried my pi with my setup, but I see how those problems can exist. I was never using the ethernet or USB duing operation, so I didn't notice those issues. I will henceforth make a modification to deal with this. I feel like I can achieve this with one electrolytic capacitor, but I will do my research before calling it a day.

I then went to the tutorial on maxbotix and saw the filter was an electrolytic cap plus two resistors. I stand corrected. Luckily I have those items at home already.

The tutorial from maxbotix is really well done: http://www.maxbotix.com/articles/074.htm
Last edited by mikey11 on Thu Feb 27, 2014 4:48 pm, edited 3 times in total.
by **mikey11** » Thu Feb 27, 2014 4:20 pm

> probably help me do it.
>
> So this will be great for my rain tank, I'll be able to have the measurements in the house and not have to go to the tank location to get updates
>
> However I have another application that I'm hoping that I may be able to use this for, but I'm not 100% that it's going to have the ability or grunt to be able to handle it. Essentially I want to build my own fill level sensor for bottling, like beer & wine etc, so it would need to be able to measure density through a glass bottle and very quickly switch off the filler as it the liquid hits the right level - here's a link to something of industrial grade http://www.peco-inspx.com/fill-level-mo ... inspection but this thing uses gamma x-ray to be able to pass through the bottle.
>
> So essentially my question is can the sensor that you've built here pass through glass and give me an accurate reading or is just not designed for this purpose?
>
> Thanks!

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

the ultrasonic can't pass through glass, it measures physical disturbances in front of it, so it would see the glass because it's in front of the liquid.

These sensors work best aiming from the top down on a liquid.
by **mikey11** » Thu Feb 27, 2014 4:21 pm

Posts: 349

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

> david4shure wrote:Just out of curiousity, did you write the Adafruit_CharLCDPlate class or does that a pre written library?

It was prewritten, and came directly from the adafruit website.

by **Tarcas** » Fri Feb 28, 2014 4:20 pm
Awesome project. Great use of the Pi for an interesting application.

Posts: 739

Joined: Thu Jan 09, 2014 5:38 am

Location: USA

> The pi is going to be set up to run for 25 minutes on a button press. This is good for the time it takes for the water guy to do his delivery. I may increase my code to have it take the initial

Is this because the battery can't power the Pi continually?
I saw one of your goals was to record the water level at one-minute intervals.
If not, might I recommend that if you implement goal 2, you can have it automatically activate the screen and higher-resolution output (start taking readings every second or two to update the screen) based on your changing water level? Meaning if your one-minute update is more than a certain percentage higher than the previous one (so that normal ripples and measurement error don't set it off) you activate the "filling mode" to turn on the screen, and maybe send you an E-mail or text to notify you that it's being filled, as well as the start level.

> The regulator on the RaspberryPi is only rated
> for 50mA maximum.

I believe this 50mA limit is for the sum of all GPIO pins. Header pins 1 (3.3v) and 2 (5v) aren't included in this maximum. I haven't had any issues powering my sensors with these (except when I accidentally shorted 3.3v pin 1 directly to GND pin 6, and it triggered a reboot. Don't do that.) However, I could be mistaken about this.
by **mikey11** » Mon Mar 03, 2014 8:41 pm
Yeah, I don't want to run continuously because the solar panel I have wouldn't be enough, and I'm not interested in spending more money on this project.

Posts: 349

Joined: Tue Jun 25, 2013 6:18 am

Location: canada

Search This Topic

Post a reply

14 posts

Return to Automation, sensing and robotics

Jump to: Automation, sensing and robo Go

Who is online
Users browsing this forum: No registered users and 21 guests

Board index

The team