



Kegminder™

A project that displays the amount of beer left in a keg

By Mark Pardue, PhD and Kelly Haupt PE

Introduction

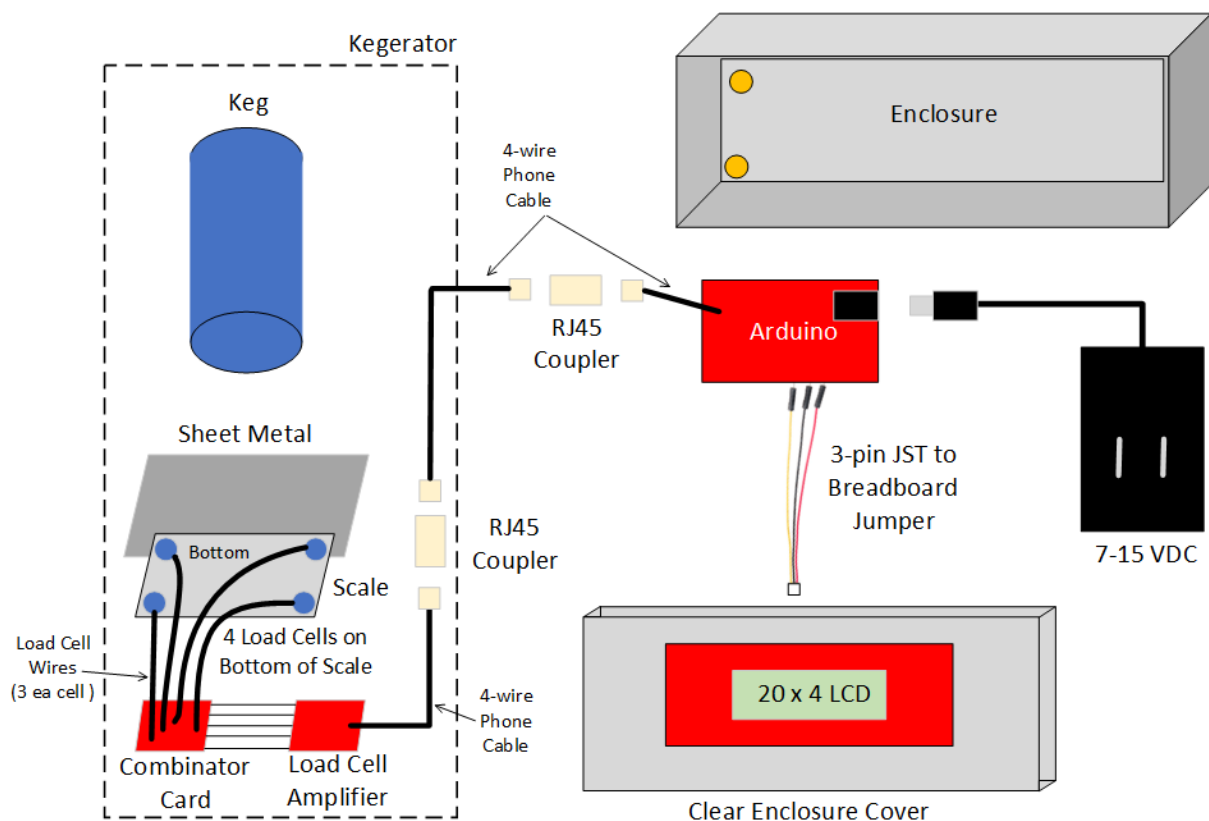
Being the owner of a Kegerator (a small refrigerator that holds a keg of beer, outfitted with a tap system), my only problem was knowing when the keg was almost empty. This system solves that problem for under \$150. My friend Kelly Haupt is a retired engineer who worked with sensors for years on Navy projects involving testing small boats, including those used by U.S. Special Forces. He is primarily responsible for the 'sensor' part of this project. Special thanks to my wife Ann for coming up with the name 'Kegminder' right off the top of her head.

Overall Design

Figure 1 shows the overall design for the Kegminder. There are two major modules:

- Sensor Module that weighs the keg and is placed inside the kegerator
- Computer & Display Module that is packed in an enclosure and is placed outside the kegerator

The two modules are connected by 4-conductor telephone cable. You should note that the way I ran the telephone cable was to have the ½ cable that was soldered to the sensor module stay completely inside



NTS

Figure 1. Kegminder Top-Level Design

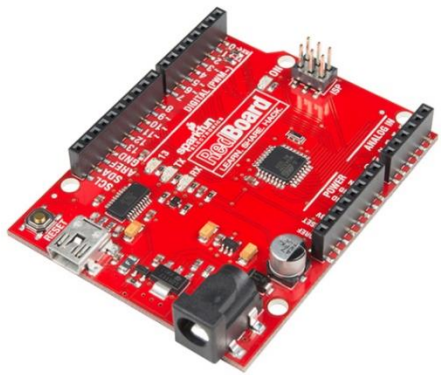
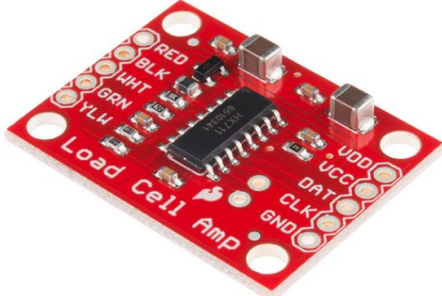


the kegerator and the ½ cable that was connected to the headers on the Arduino board (in the computer & display module) stay completely outside the kegerator. That allows for easy maintenance, since you can remove either the scale (sensor module) or the outside electronic enclosure (containing the computer & display module), and not have to pull any cables through the kegerator walls.

The ½ of the telephone cable that is connected to the Arduino board needs to be solid conductor to allow easy insertion in the headers. Also using solid conductor for the ½ of the cable soldered to the Load Cell Amp makes it easier to solder.

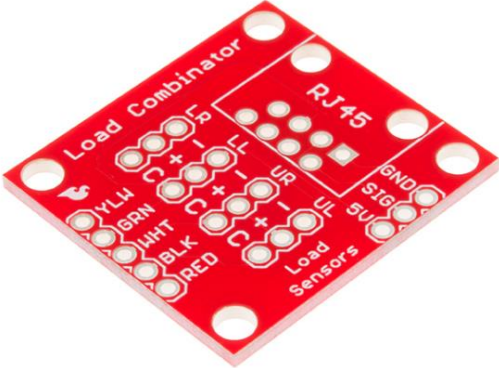


The connections between all 3 of the telephone cables were made using RJ-11 Male-to-Male connectors. *

**Some RJ-11 Male-to-Male connectors reverse the two pairs of wires connected to them. This doesn't have a big impact on telephone communication but can have devastating effects on this project design. There are telephone line testers that can tell you if the pairs have been reversed. In the case of my implementation, BOTH of the connectors reversed the pairs, which happily resulted in providing the correct connections.*

Materials we Used

<i>Purchased from www.sparkfun.com</i>		
Arduino SparkFun RedBoard - Programmed with Arduino DEV-13975	 A red Arduino SparkFun RedBoard with a USB Type-B port, a DC power jack, and various pin headers.	\$19.95
Load Cell Amplifier HX711	 A small red PCB with a central black chip, labeled 'Load Cell Amp'. It has several pins labeled 'RED', 'BLK', 'WHT', 'GRN', 'YLU', 'VDD', 'VCC', 'DAT', 'CLK', and 'GND'.	\$9.95
20 x 4 LCD with Backpack Serial Enabled 20x4 LCD - Black on Green 5V LCD-09568*	 A red PCB with a 20x4 LCD display showing 'Hello World!'. It has a black frame and a red PCB with several pins.	\$29.95
3-pin JST to Breadboard Jumper CAB-13685	 A small PCB with three wires (red, yellow, black) and a JST connector.	\$1.50

**Something to note: This LCD display might be a bit 'delicate.' Some reviewers have said it failed after a few days, and I had one of these fail as was putting it in an enclosure.*

<p>Combinator Card SparkFun Load Sensor Combinator BOB-13878</p>		<p>\$1.95</p>
<p>Jumper Wire - PTH Black White PRT- 08672 (I soldered both ends to provide 5VDC to 2 connections)</p>		<p>\$0.95</p>
<p><u>Purchased from www.alliedelec.com</u></p>		
<p>Enclosure purchased from Hammond Manufacturing 1554J2GYCL Polycarbonate; UL945V; Gray; 6.3x3.5x2.4 In; NEMA13; 1554 Series, with clear cover and seal. Larger than required, others were too small.</p>		<p>\$21.13</p>

<u>Purchased separately from Other Sources</u>		
Standard Digital Bathroom Scale		\$19.95
Telephone wires with RJ-11 plugs, 4-wire solid conductor (2 req'd)		\$5.00
RJ-11 Male-Male connectors (2 req'd)		\$3.00
9-15 VDC Power Supply		\$10.00
120V Switch (allows for easy on/off operation)		\$5.00
Extra Jumper Wires		\$3.00
Sheet Metal* approx. 14" x 14"		Scrap

**I used the side panel of an old PC and bent a lip to keep it from slipping off of the scale. Digital scales are a bit too fragile to have a keg put directly on top of them, so this seemed to distribute the pressure a bit more evenly.*

Sensor Module

Digital Scales have 4 load cells, one situated at each corner of the scale, and each load cell has 3 wires. On my scale, they were color coded:

- Red = Common
- White = +
- Black = -

No matter what the color coding on the wires, the largest resistance is always the + to – wires. For our project, Kelly used a special Combinator circuit board that helps simplify the wiring. The Sparkfun circuit takes the 12 wires from the 4 load cells on the scale, and connects them in a Wheatstone bridge circuit, resulting in only 5 output wires.

You feed these 5 output wires to the Sparkfun Load Cell Amplifier Circuit Board. On the Load Cell Amp, we soldered VCC and VDD together since we were using 5VDC for both. That way, we only needed to run 4 wires from the Amp to the Arduino board. We used standard 4-wire solid telephone cable with an RJ-11 connector for that run. As discussed above, an easy way to do this is to take a spare 4-conductor solid wire telephone cable, cut it in half, and save the other half for the connection to the Arduino board.

I taped the Combinator circuit board and the Load Cell Amp circuit board to the bottom of the scale (see Figure 2) *. Since the 4 ‘feet’ of the scale (where the load cells are located) provide a bit of vertical clearance, the circuit boards are a protected.

**Disclaimer: I have no idea what prolonged temperatures of close to freezing will have on the Load Cell Amp, but at least you don't have to do any temperature compensation circuitry, because the temperature should be constant inside the kegerator.*



Figure 2. Sensor Module

Computer & Display Module

The Computer & Display Module contains two circuit boards

- Arduino board
- LCD display

It also has 2 cables coming in:

- 4-conductor telephone cable from the Sensor Module
- 7-15 VDC power supply connection

I chose the smallest enclosure I could find to hold the two boards, and it still had plenty of room. The key to the enclosure is to have a **CLEAR COVER** so the LCD display shows through. The enclosure I chose also had a rubber gasket to help with dust and moisture. Figure 3 shows the enclosure and some of the connections.

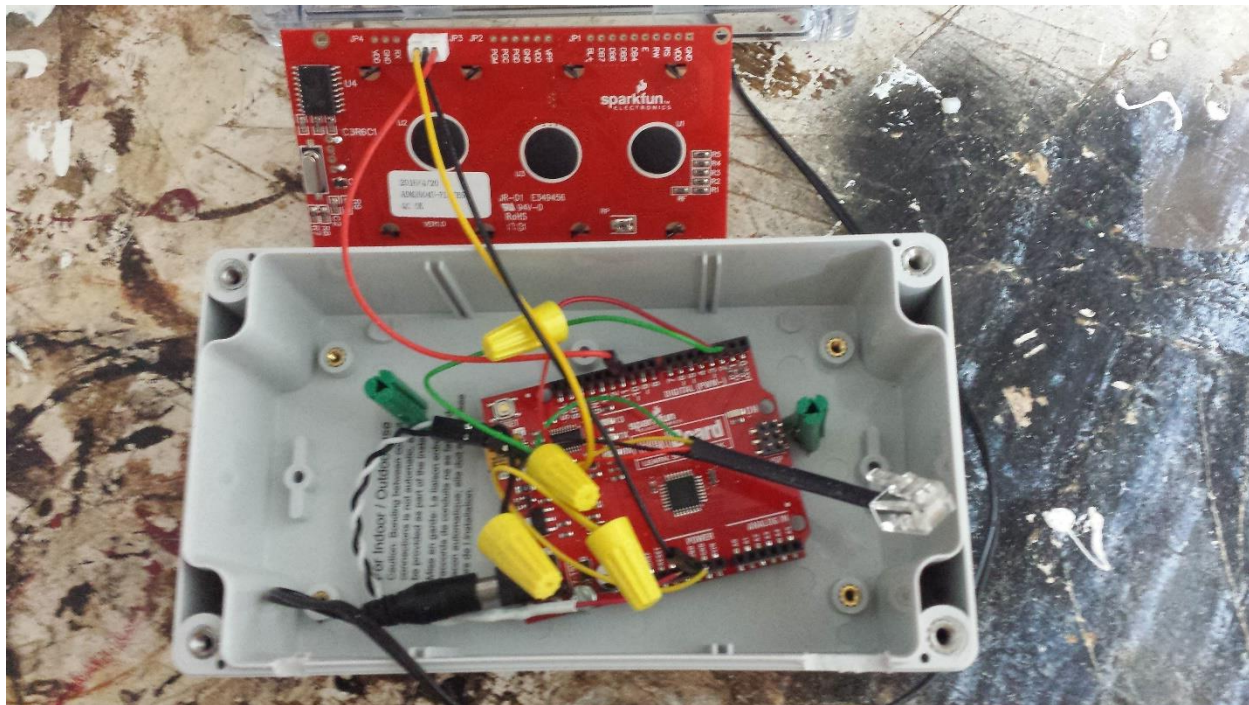


Figure 3. Computer & Display Module

I used wire nuts to connect some of the leads from the telephone cable (but soldered the wires together first, to prove a better physical connection also. We needed a 5V connection to send out of the telephone cable to the sensor module and another 5V connection to the LCD display. Since there was only one 5V header, I used an old PC connector with a white wire and a black wire (shown in Figure 3), which terminated in a 2- connection female connector. By soldering the black and white wires together, I was able to provide 2 connectors, both with 5VDC for the two required connections.

I used a small rotary tool to grind down the side of the enclosure for all the two cables to enter the enclosure. *

There are slots in this particular enclosure to place circuit boards vertically, but that wouldn't work for this application. The LCD display needs to be horizontal, and the Arduino circuit board isn't the correct size to be mounted vertically in the enclosure, so I had to 'MacGyver' the installation a bit. Only one mounting standoff matched the mounting holds on the Arduino board, but by mounting it at an angle the other side of the board it was secured in a stable manner. For the LCD display, I used the two green cylinders which are cylindrical drywall anchors, shown in Figure 3. I screwed these in through the bottom of the enclosure. The front edge of the LCD display rests on the vertical circuit board slots. Since the green sheet rock anchors are a bit shorter than the vertical slots, the LCD angles downward at the bottom, which is perfect for this application, since you don't have to view the enclosure from directly above to read the display.

**A 'production' version of this system would have a hole on the outside of the enclosure where the barrel jack of the power supply would directly connect to the flush-mounted board and would also incorporate a flush-mounted RJ-11 jack on the outside of the enclosure.*

Software

I programmed the Arduino board in the C language, using the functions that the Arduino understands. The IDE and software libraries are all identified, with instructions, on the Sparkfun site. The source code I developed is at Appendix A and is (I think) fully commented. Two libraries, from other contributors, are required:

- SoftwareSerial.h for the serial-enabled LCD*
- HX711.h for reading the Load Cell Amp output from the scale

I chose to keep the percent keg full value as an integer, since the accuracy of this system doesn't really support decimal places. The accuracy of the system appears to be 5%, give or take. The raw scale reading for full kegs and empty kegs varies about 5%. I took a number of readings for several kegs over a period of time and used the average of those readings in the software. In the source code, the sections I used for getting those preliminary readings are commented as 'Debugging' so you can uncomment those sections and to your own readings.

**If you use a standard LCD, you will need many more connections (soldered) than the 3 that the JST connector provides and will need to use a different library file.*

Appendix A: Source Code

```
/*
  Uses the SparkFun HX711 breakout board with a digital scale
  By: Mark Pardue
  Date: Aug 2, 2019
  License: This code is public domain but you buy me a beer if you use this and we meet
           someday (Beerware license).
*/

#include "HX711.h"
#include <SoftwareSerial.h>

#define calibration_factor 7050.0 // This value was provided by the author of the HX711.h
                                   // library using the SparkFun_HX711_Calibration sketch

#define DOUT 3 // Data Out
#define CLK 2 // HX711 CLK

// These are the average of values obtained during multiple debugging runs
#define EMPTY 850 // Empty keg scale reading
#define FULL 1918 // Full keg scale reading

long range = FULL - EMPTY; // Used to calculate percent left

HX711 scale(DOUT, CLK); //Setup HX711 to scale

// Attach the serial enabled LCD's RX line to digital pin 11
SoftwareSerial LCD(10, 11); // Arduino SS_RX = pin 10 (unused), Arduino SS_TX = pin 11

void initDisplay();
```

```
void setup() {

    LCD.begin(9600); // set up serial port for 9600 baud
    delay(500); // wait for display to boot up

    scale.set_scale(calibration_factor);

    //Initialize Display
    initDisplay();
}

// This function clears the display and prints the first two 'boilerplate' lines
void initDisplay()
{

    // move cursor to beginning of first line
    LCD.write(254);
    LCD.write(128);

    // clear display by sending spaces
    LCD.write(" ");
    LCD.write(" ");
    LCD.write(" ");
    LCD.write(" ");

    // move cursor to beginning of first line
    LCD.write(254);
    LCD.write(128);

    LCD.write(" Kegminder v5 2019");
    LCD.write(" ");
```

```

LCD.write(254);
LCD.write(14);

LCD.write(" M.Pardue/K.Haupt ");
}

void loop()
{
  long scaleValue;    // Raw number from scale
  int perValue;      // Percent left, no decimal places
  int bars;          // Number of bars to display current percent left
  char valueText[9]; // Temporary string to convert numbers to display

  // Get the reading from the scale
  scaleValue = scale.get_units()*10;

  /*
  // For debugging: Uncomment this section and comment out section noted below
  // Display the scale's raw number
  ltoa(scaleValue, valueText, 10);
  LCD.write(valueText[0]);
  LCD.write(valueText[1]);
  LCD.write(valueText[2]);
  LCD.write(valueText[3]);
  */

  // For debugging: Comment this out if debugging to adjust display
  // Add 4 blanks to display if NOT debugging
  LCD.write("    ");

```

```
// Check for hardware error
if(scaleValue < 0)
{
    scaleValue = 0;
}

// Calculate percent
perValue = ((scaleValue - EMPTY) * 100) / range;

//Make sure numbers between 0 and 100
if(perValue > 100)
{
    perValue = 100;
}
else
{
    if(perValue < 0)
    {
        perValue = 0;
    }
}

// Convert percent to string
ltoa(perValue, valueText, 10);

// Add blanks to center percent number
LCD.write("    ");
```

```
//Figure out how many digits to write
if(perValue < 10) // Single digit
{
    LCD.write(" ");
    LCD.write(valueText[0]);
}
else
{
    if(perValue < 100) // 2 digits
    {
        LCD.write(" ");
        LCD.write(valueText[0]);
        LCD.write(valueText[1]);
    }
    else
    {
        if(perValue == 100) // 3 digits
        {
            LCD.write(valueText[0]);
            LCD.write(valueText[1]);
            LCD.write(valueText[2]);
        }
    }
}

// Put the percent sign on the end
LCD.write("%");

// Finish the line
LCD.write(" ");
```



```
// To get bars on display
// How many bars on the bottom row?
bars = perValue / 5; // 20 characters on the display = 100%

if(bars > 0)
{
    for (int i = 1; i <= bars; ++i)
    {
        LCD.write(0xFF);
    }
}

delay(5000); // Wait 5 seconds then refresh entire display

initDisplay();
}
```