7-25-2015

Open Source Electronics for Laboratory Physics

Zengqiang John Liu
St. Cloud State University, zliu@stcloudstate.edu

Open Source Electronics for Laboratory Physics

Dr. John Liu

Saint Cloud State University, MN, USA
zliu@stcloudstate.edu

AAPT workshop W12 July 25, 2015

Acknowledgement

- Saint Cloud State University, MN
- Vernier Software and Technology
- My family (for putting up with my weekend, holiday and summer work and all electronic components at home)
- Students: Cheng Xu, John Olson, Frank Leo, Anthony Walz, Akan Essien

Agenda

- CH1: Introduction to open source electronics
- CH2: Sensors
- CH3: Basics of programming
- CH4: Open source physics lab device discussion
- CH5: Additional OSPL features
- 10 minutes at the end of each hour for breaks and soldering practice

Challenge!

Learn laboratory physics skills in 40 hours!

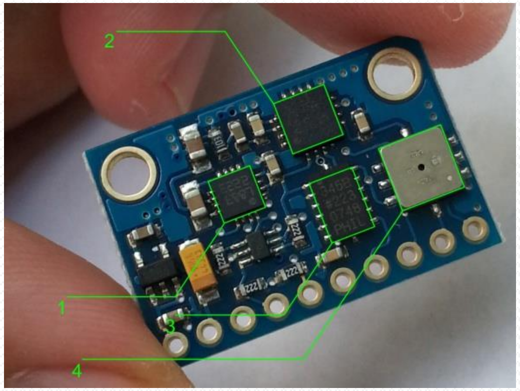
Restrictions:

- You can’t use your current knowledge level. Start at level 1.
- You can only practice your skills at 2 hours a week.
- Your hours are fixed, you show up whether you like it or not.
- If you get stuck, or you come unprepared, you can’t pause.
- If you made a mistake, fix it within the 2 hour limit, hurry!
- There is only one instructor, your clock ticks while you wait for an answer.
- You can try to ask other students but they are just as busy.
- Next week you spend 2 hours on a different topic.
- Good luck! You will need it sometimes.

CH1: Open source electronics

- Electronics here represents circuit board designs
- Designers release design files under public licenses
- Global online forums support new and experienced users
- Dramatically reduces cost on lab data acquisition systems
- Users may modify designs to meet their needs
- Dramatically expands sensor selection

3-axis accelerometer, 3-axis magnetic sensor, 3-axis gyroscope, barometer
Paid $13, requires 4 wires.

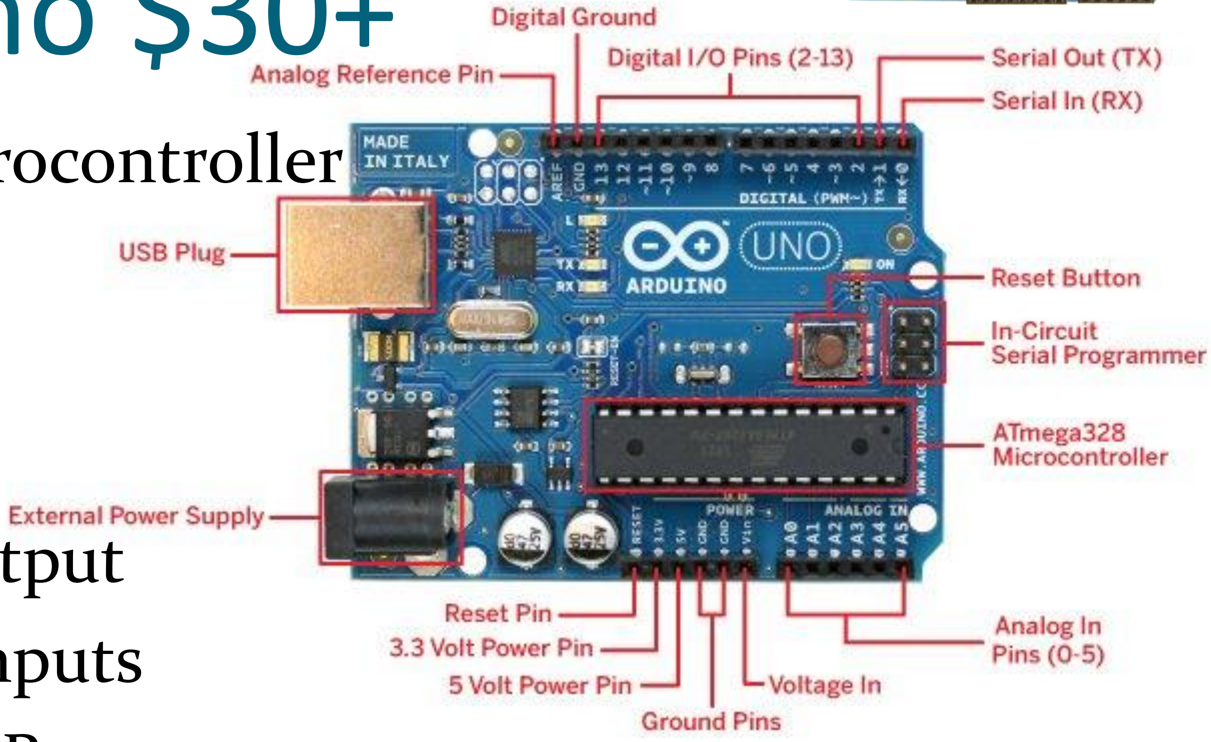


Example open source electronics

- Arduino: microcontroller development platform
- Beagle board, Raspberry pi: open source single-board GNU/Linux computer
- Open source physics laboratory platform: lab data acquisition system based on Arduino
- RepRap: open source 3D printer
- Openmoko: open source mobile phone

Arduino Uno $30+

- 16MHz 8-bit microcontroller
- 2KB SRAM
- 1KB EEPROM
- 32KB FLASH
- 20 total input/output
- 6 10-bit Analog inputs
- Serial port via USB
- SPI interface (SD card, Wi-Fi, Ethernet etc.)
- I²C (real time clock, accelerometer, gyroscope etc.)



Google arduino project:

640 × 480 - wired.com
IT'S TWENTY QUARTER FIVE MINUTES TEN O'CLOCK DOUG'SARDUINO
Secret Knock Detecting Lock
Parts:
http://arduino-guay.blogspot.com
Arduino PONG
2387 2400
THE ENOUGH ALREADY
Make

Arduino shields

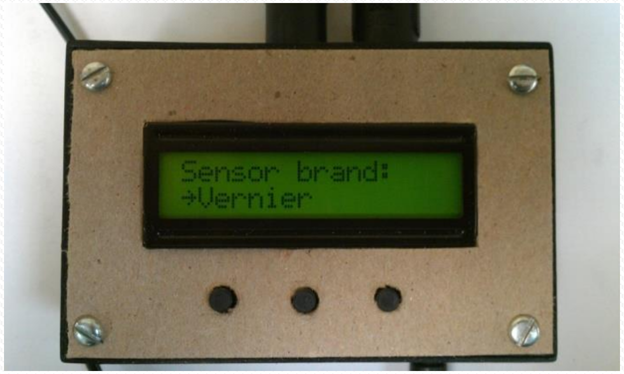
- Sits on top of an arduino
- Adds functionality to arduino, such as Ethernet, Wi-Fi, SD card, xbee, real time clock, motor control, display, GPS, GSM, Bluetooth, RFID, MIDI, VGA, TV etc.
- Many shields are designed by 3rd party

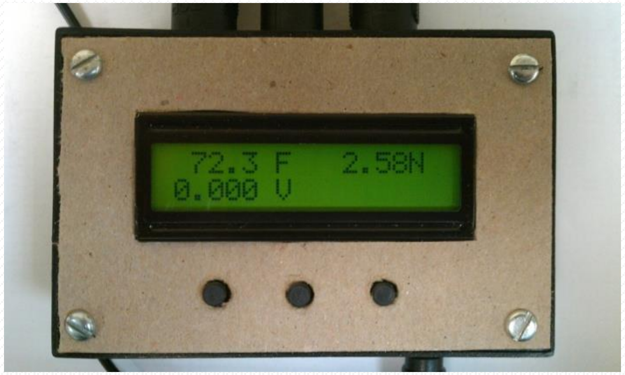
Open source physics lab

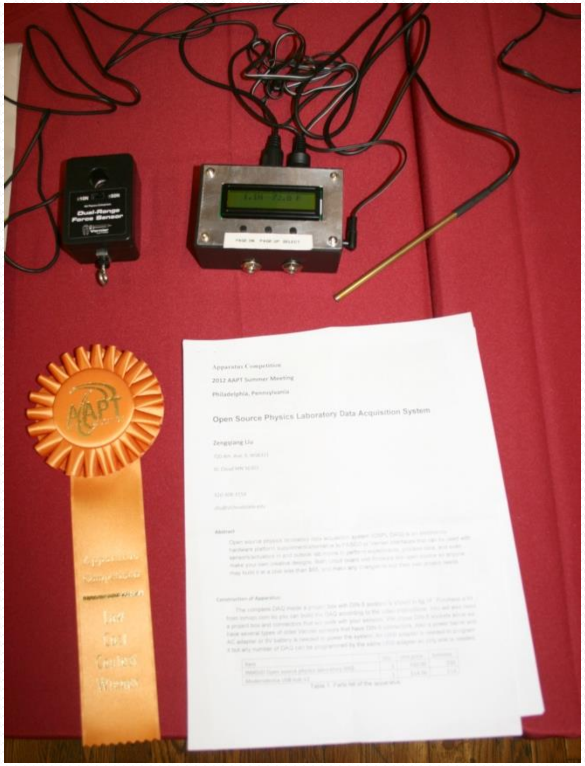
- Based on Arduino ecosystem with huge improvements
- Runs Arduino code and compatible with many Arduino hardware and sensors
- Rugged design to survive student lab environment
- Micro-SD card and real-time clock for data logging
- LCD and rotary encoder for easy user interface
- No loose wires or exposed circuits on breadboards
- Compatible with Vernier and other sensors
- Compatible with more modern sensors
- Twice award winner at AAPT apparatus competition

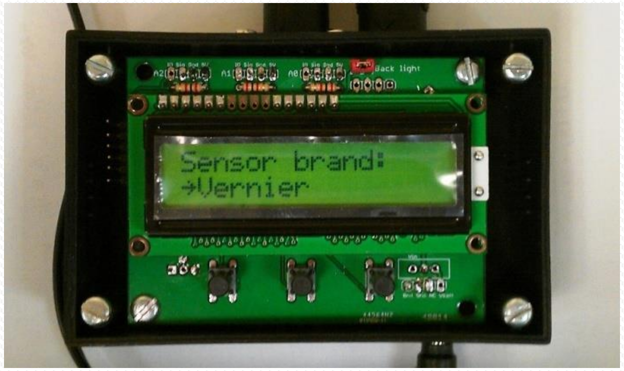
OSPL 1.0 (2012)

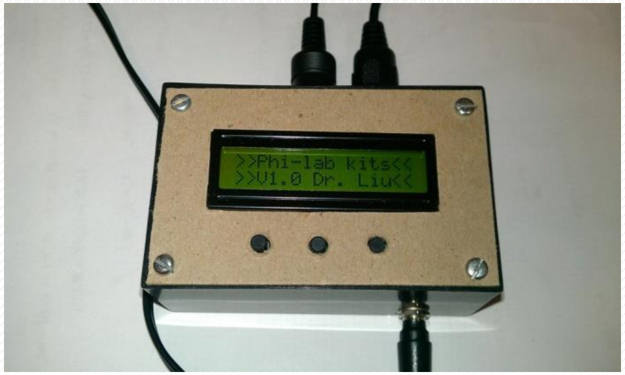
AAPT 2012



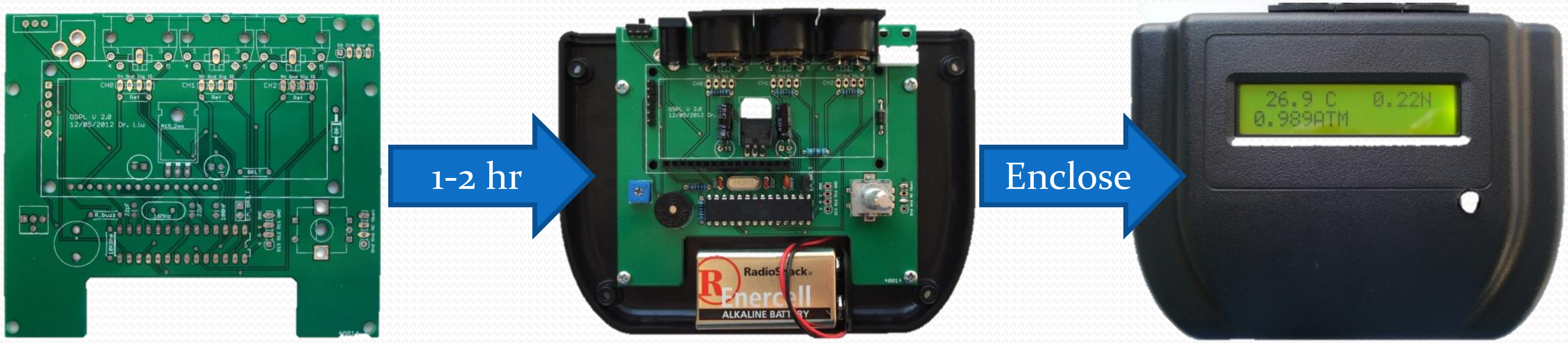



Dual-Range Force Sensor
AAPT
Apparatus Competition
2012 AAPT Summer Meeting
Philadelphia, Pennsylvania
Open Source Physics Laboratory Data Acquisition System
Zengqiang Liu
Abstract
Construction of Apparatus

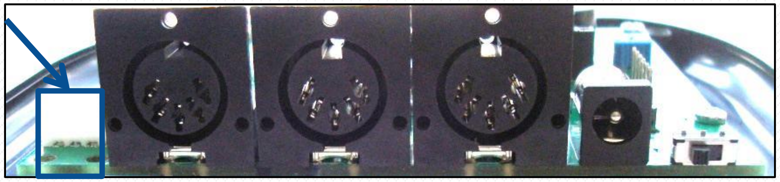




OSPL 2.0 (2013)



DIN-5 connectors, power barrel, on/off, USB port, Bluetooth (internal)

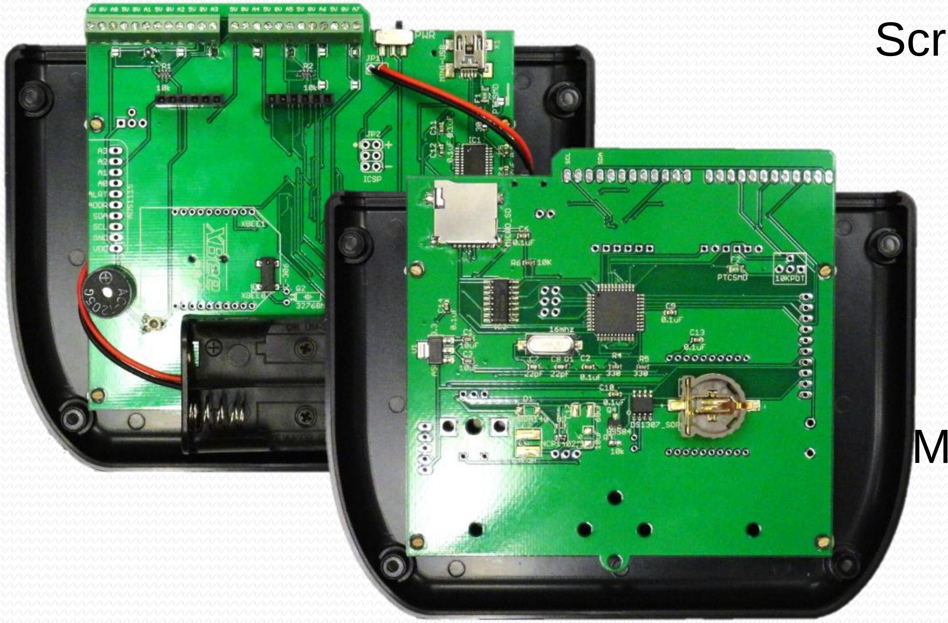


DIN-5 plugs won’t fall off like jumper wires on an Arduino!

AAPT 2013
Apparatus competition
award and low cost winner

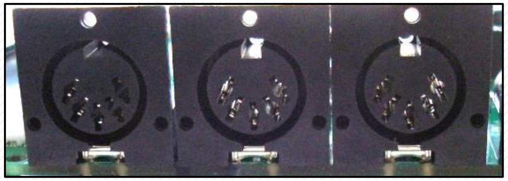
OSPL 2.4 (2015)

- Improved integration with ATMEGA1284 processor
- Xbee wireless module socket for Wi-Fi, Bluetooth etc.
- SD card Real-time clock make data logging accurate
- 2 AA batteries and option to sleep to conserve power
- Optional 4-channel 16-bit analog-to-digital converter



Screw terminal blocks or DIN-5 sockets

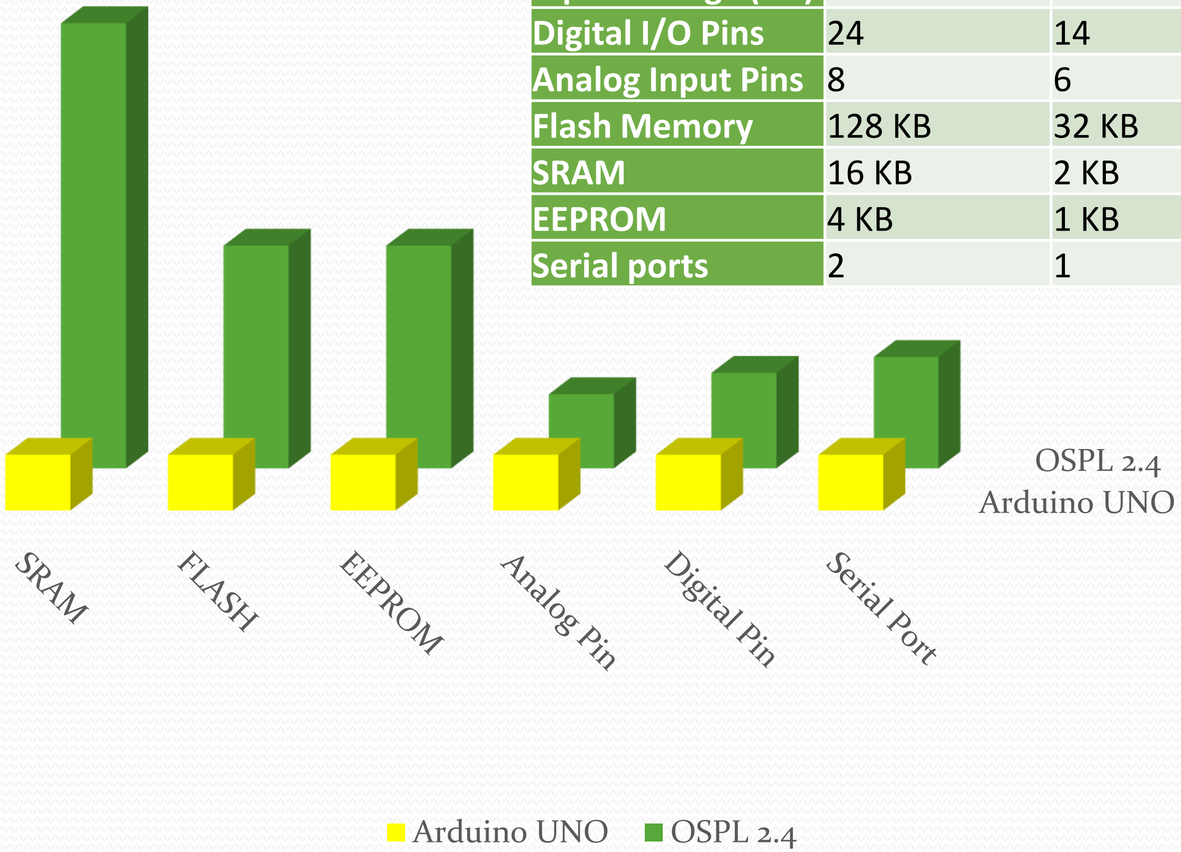
5V 0V A0 5V 0V A1 5V 0V A2 5V 0V A3
5V 0V A4 5V 0V A5 5V 0V A6 5V 0V A7



Micro-SD card, ON/OFF switch mini-USB (Not shown)

Arduino UNO vs. OSPL V 2.4

Summary	OSPL V 2.4	Arduino UNO
Microcontroller	ATmega1284P	ATmega328
Input Voltage (DC)	2 AA batteries	Power barrel
Digital I/O Pins	24	14
Analog Input Pins	8	6
Flash Memory	128 KB	32 KB
SRAM	16 KB	2 KB
EEPROM	4 KB	1 KB
Serial ports	2	1



Firmware:

- Standard firmware is compatible with many sensors
- Log data to SD card by turning the knob to the right
- Pause reading by turning the knob to the left
- Many Vernier sensors can be selected from a list.
- Sensors not on the list requires a simple conversion.
- Read and voltages and resistance values
- I^2C sensors include 16-bit analog-to-digital converter, 3-axis accelerometer, 3-axis magnetic sensor, 3-axis gyroscope, barometer, ambient temperature sensor
- Adjust logging interval, date and time for data logging

Firmware menu

Analog Sensor

Vernier
- Direct Temp
- Temperature
- Force +-10N
- Force +-50N
- Pressure PS-DIN
- Gas Pressure
- Voltage
- AUTO_ID
- Cond. 200uS/cm
- Cond. 2KuS/cm
- Cond. 20KuS/cm
- Accelerometer
- Low-g Acce.
- Hand Dynamom.
- Light (6K lux)
- Magnetic (10X)

OSPL
- Voltage (Even)
- Resistance
- 10KΩ Thermistor
- Voltage (Odd)
- TMP36 temp.

No sensor

Digital Sensor

- Vernier Ranger
- OSPL Ranger
- Photogate
- DS18B20 Temp

I2C

- ADXL345 Accele
- BMP083 Baromet
- HMC5883L maget
- ADS1115 ADC

Settings

- UI delay
- Ser./SD delay
- Set clock
- LCD back light
- Melody
- Show credits

While in menu:

Cycle through menu

Select

While reading data:

Pause|Live|Log to SD

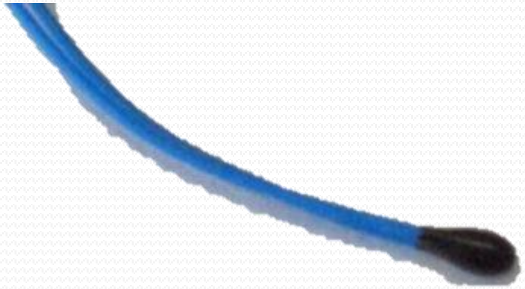
Return

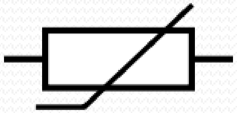
CH2: Sensors

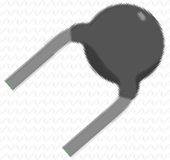
- Resistive sensors (thermistors, photo resistors etc.)
- Analog sensors (Vernier and other analog sensors output voltage representing measurement)
- Digital on-off sensors (photo-gates, Hall-effect switches, drop counters etc.)
- Digital pulse width sensors (sonic rangers)
- Digital sensors with serial output (some sonic rangers)
- Digital sensors with I^2C bus (accelerometers, magnetic sensors, gyroscopes etc.)
- Digital sensors with SPI, One-Wire, SDI-12 interfaces

Resistive sensors

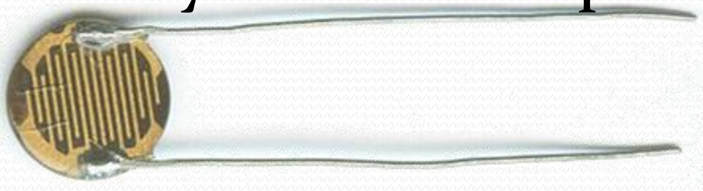
- Resistance represents physical quantities
- Thermistor: doped semiconductor that decreases resistance with increasing temperature.

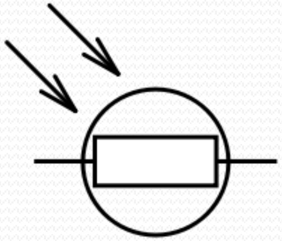




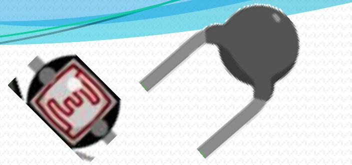


- Light dependent resistor (LDR): doped semiconductor that decreases resistance with increasing light intensity. Also called photo resistors.



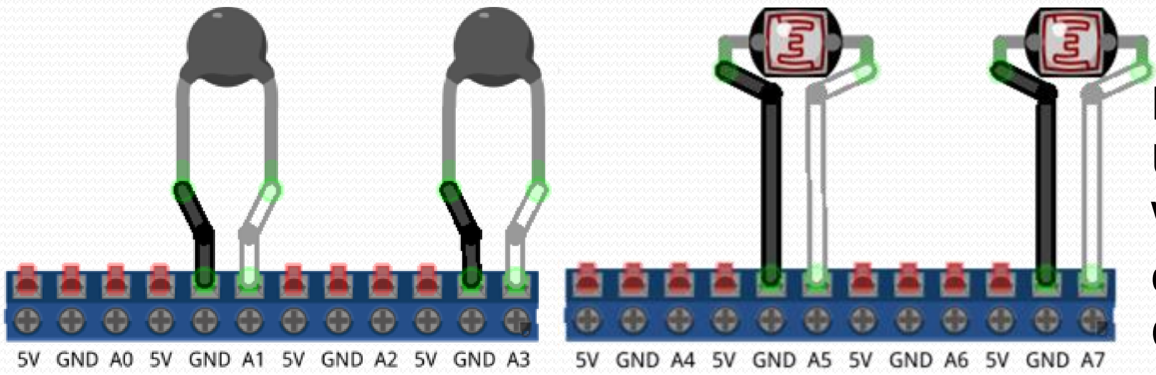


Wikipedia.org



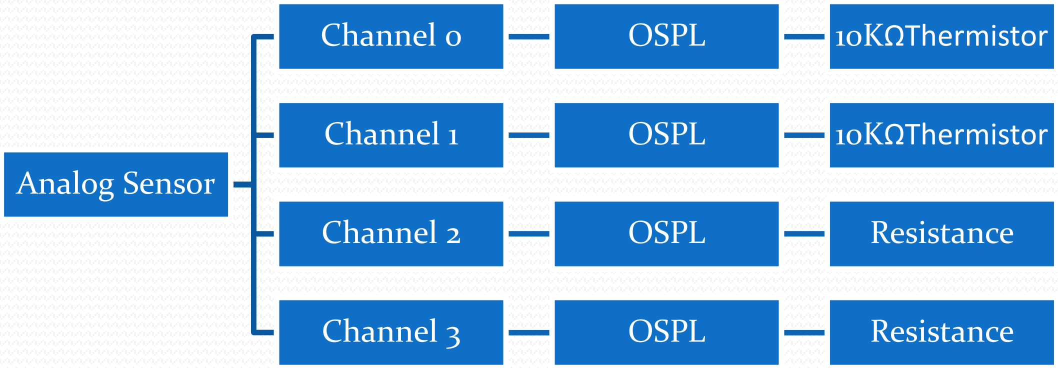
Thermistors and photo resistors:

Connection diagram:


5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

Dev. edition:
Use any pin with 10KΩ pull-up
Wiring:
one end – GND
Other end – A1, A3, A5, or A7

Firmware menu:

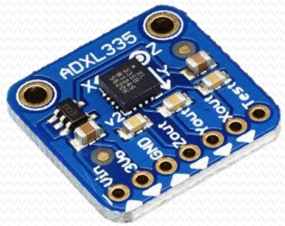


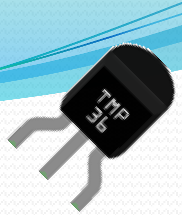
Analog Sensor
- Channel 0 — OSPL — 10KΩThermistor
- Channel 1 — OSPL — 10KΩThermistor
- Channel 2 — OSPL — Resistance
- Channel 3 — OSPL — Resistance

Tip:
Try “Voltage (Odd)” or “Resistance” for thermistor.
Try “Voltage (Odd)” for the photo resistor.

Analog sensors

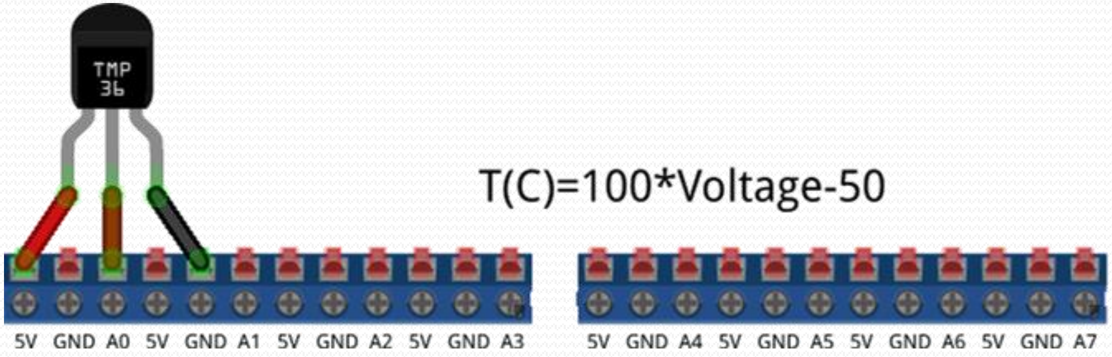
- The sensor usually requires power (5V and GND)
- The sensor has a pin that outputs a voltage
- The voltage is linear to the measurement
- Measurement = Slope * Voltage + Intercept
- Vernier direct temperature probe (DCT-DIN)
- Slope 55.55°C/V or 100°F/V
- Intercept -17.7°C or 0°F
- Some analog sensor reading is *ratiometric*, meaning the output voltage is proportional to supply voltage.





TMP36 temperature sensor:

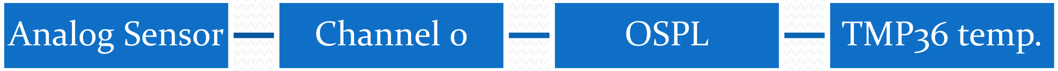
Connection diagram:



T(C)=100*Voltage-50

Dev. edition:
Use any pin W/O10KΩ pull-up
Wiring:
Face marking forward
Left pin – 5V
Middle pin – A0, A2, A4, or A6
Right pin – GND

Firmware menu:

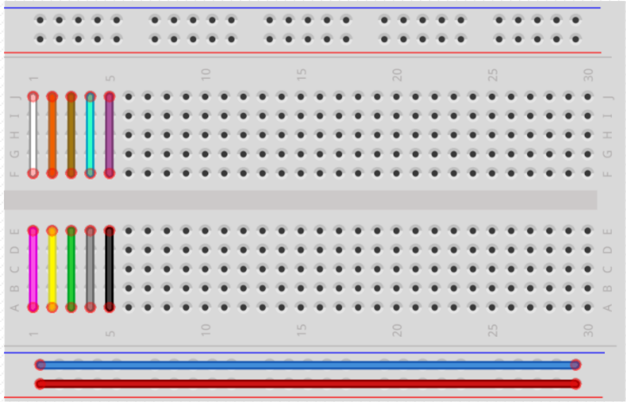


Analog Sensor — Channel 0 — OSPL — TMP36 temp.

Tip:
Try “Voltage (Even)” and find out the voltage-temperature relation.

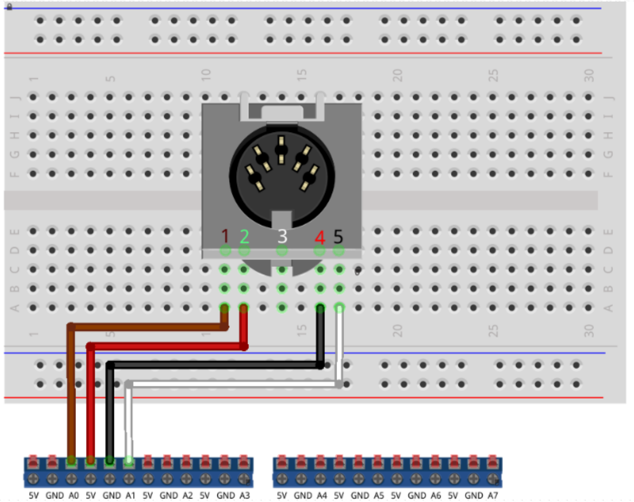
Breadboard

- Columns are marked with letters A-J.
- Rows are marked with numbers 1-30 or more.
- All 5 holes in a row are connected and they act as junctions you see on a schematic.
- All holes along a red bus are connected but different red buses are separate. Same goes with blue buses.



Wiring a Vernier analog sensor

Connection diagram:


1
2
3
4
5
5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

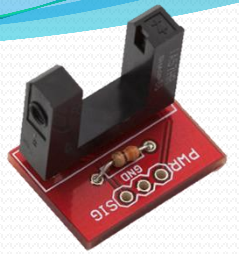
Dev. edition: Use any analog pin **without** 10Kohm pull-up resistor (A0, A2, A4, A6)
Wiring: socket facing away, first pin in D11
Pin1: A0 (or any of A0, A2, A4, A6)
Pin2: 5V
Pin4: GND
Optional pin5: A1(or any of A1, A3, A5, A7)

Firmware menu:

Analog Sensor — Channel 0 — Vernier

Tip:
Try “Analog->Vernier->Voltage”
Does the voltage make sense?
Read slide #42 for slopes and intercepts for your sensor.

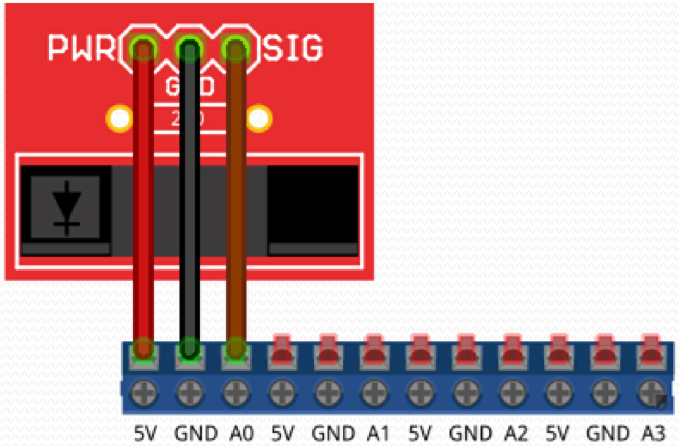
Digital ON-OFF sensor



- They output ON or OFF states
- ON/OFF state or transition represents measurement
- A transmission photogate is ON until blocked (OFF)
- A reflective photogate is OFF until object reflects light into its sensor (ON)
- To sense rotation, divide pulse/sec by # of spoke
- To sense speed, use two blockers at fixed distance
- To sense a photogate to 1cm/s speed using 1cm blockers, we need millisecond timing accuracy.

Photo-interrupter:

Connection diagram:

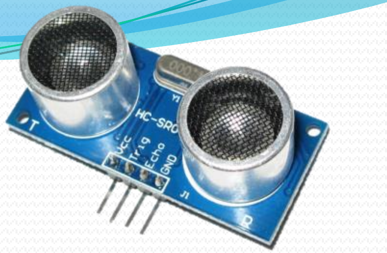


Firmware menu:

Digital Sensor — Photogate

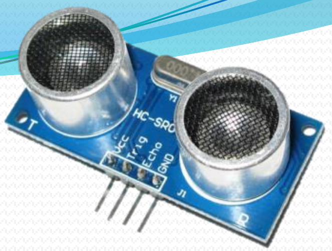
Dev. edition:
Use any analog pin
Wiring:
Face pins towards data logger
Left pin (PWR) – 5V
Middle pin (GND) – GND
Right pin (SIG) – A0 (or 1~7)

Tip:
How many sheets of paper is needed to block the IR beam?
Try “Analog->OSPL->Voltage (Even)”.
What voltage is HIGH(LOW)?



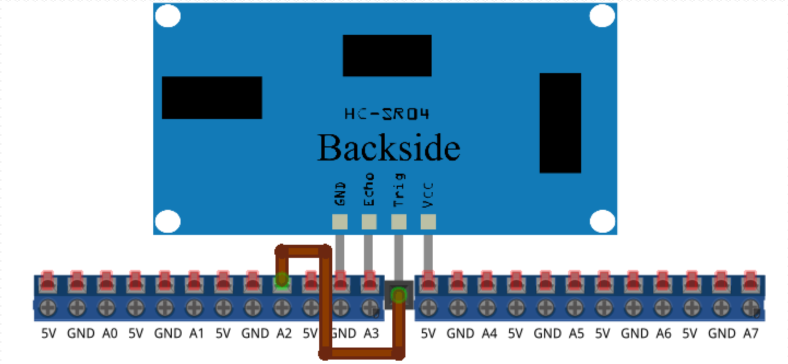
Pulse-width sensor

- They output a pulse OFF-ON-OFF
- The length of the ON portion represents measurement
- Start ranging by pulling the trigger pin to HIGH for 10us or longer then return it to LOW
- Ranger will emit ultrasonic pulses and detect its echo
- Ranger pulls the echo pin HIGH for the amount of time it takes sound to travel the distance round trip
- Detect this pulse width with pulseIn() and calculate distance with speed of sound.
- Accuracy depends on timing accuracy of the receiver.



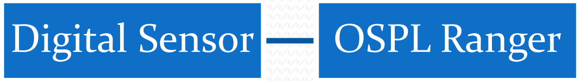
Sonic ranger

Connection diagram:


HC-SR04
Backside
GND
Echo
Trig
VCC
5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

Connect a male-female wire (brown) between “Trig” and A2 and just insert the ranger facing away from the unit into GND, A3, and 5V as pictured. Tighten the pins with screw driver.

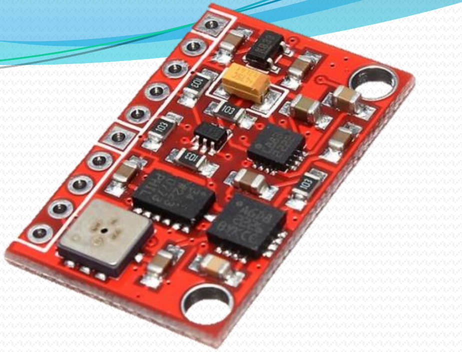
Firmware menu:



Digital Sensor — OSPL Ranger

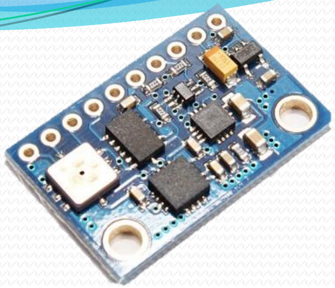
Tip:
Use the ruler to test the accuracy of the ranger.
What is its max(min) range?

I^2C sensors

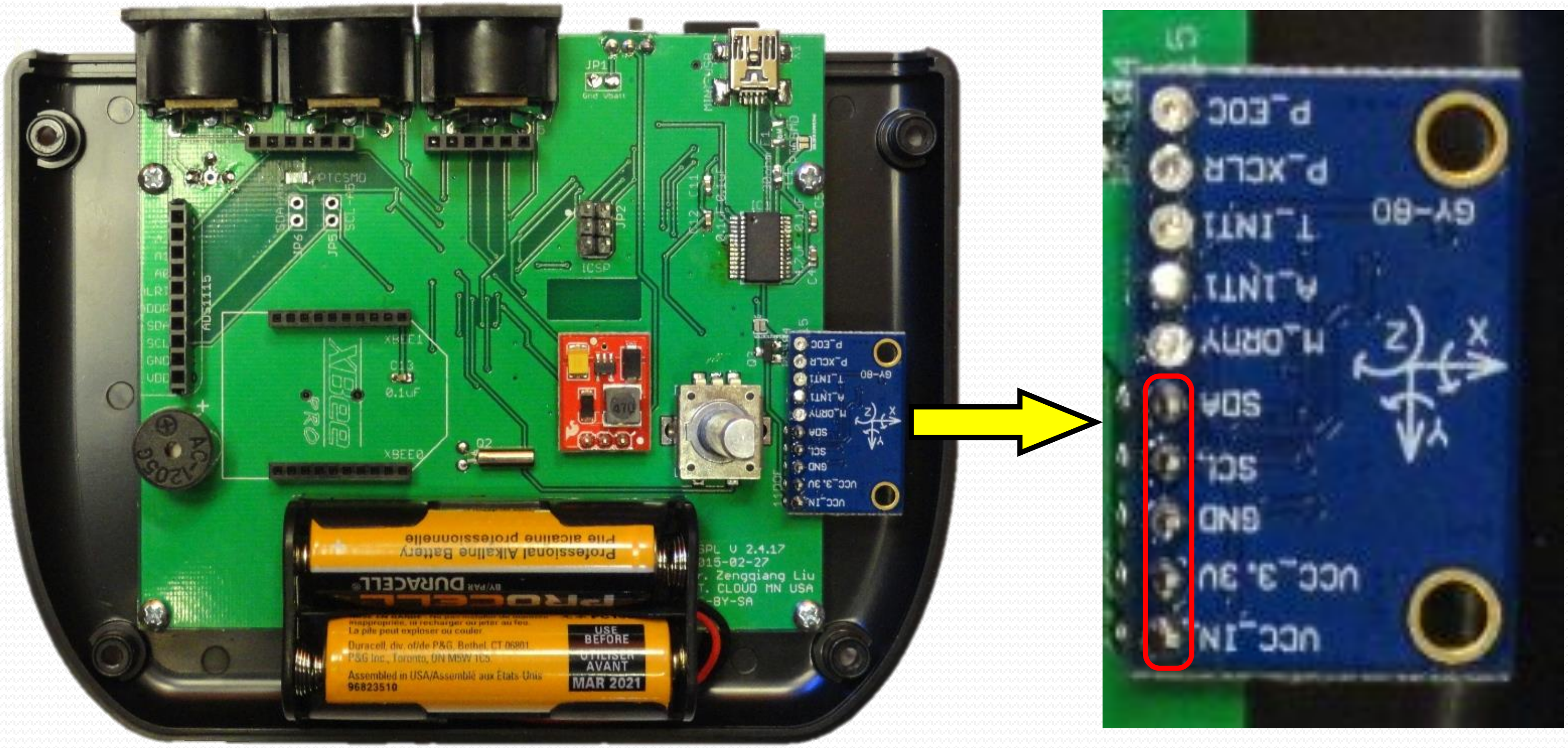


- Inter-Integrated Circuit bus connects to many sensors.
- Two wires for a large number (127) of different sensors
- Each sensor module has a different address
- Measurements are done on board and suffers NO loss of accuracy during transmission.
- Requires the Wire library to communicate with and some understanding of the commands on data sheets.

10DOF board wiring



- Solder the board upside down to the data logger

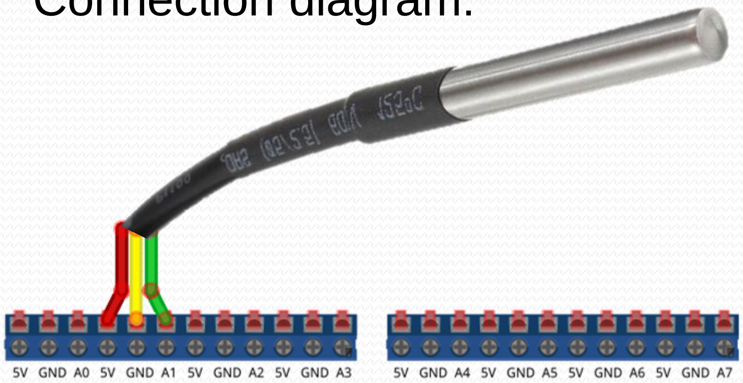


10DOF board Activities

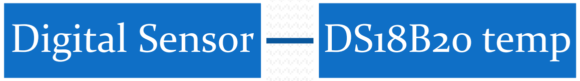
- All menu options are in I2C (don’t select ADS1115)
- Hold the device at different orientations and observe accelerometer reading
- Hold the device at different headings and observe magnetometer reading
- Hold the device at different height and observe pressure sensor readings (Tip: you can tell 0.5m height change)

One-wire sensor: DS18B20

Connection diagram:


5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

Firmware menu:



Digital Sensor — DS18B20 temp

One-wire protocol was designed by Dallas Semiconductor and can literally run on one wire (parasitic) if needed: the sensor and logger are both connected to the ground and have one wire, the data wire in between. The logger pulls the data wire HIGH for long enough in order for the sensor to receive just enough power to make a measurement and report back.

Tip:
This sensor is waterproof.
Do you know how much such a sensor might cost?

A: $3

CH3: Programming

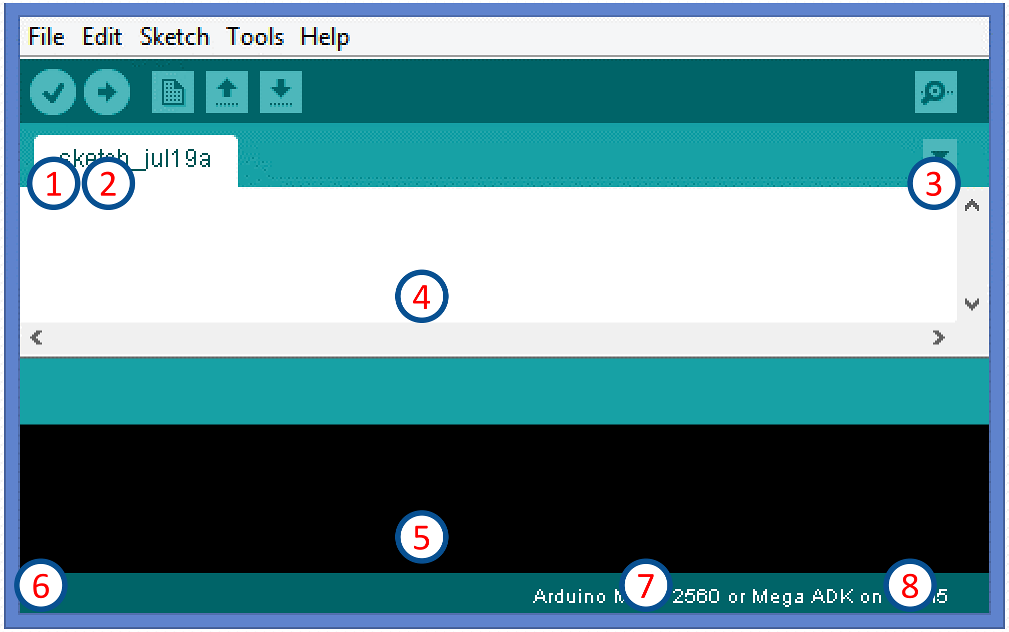
- Only very basic programming skill is needed, which you will learn during the workshop.
- C/C++ is used in Arduino. It is versatile and powerful. Arduino IDE made it easy by hiding some details.
- You type your program in an editor (IDE),compile then upload to Arduino or OSPL that runs the program.
- New upload erase previous upload.
- Some math expressions need transcription and explicit multiplication signs.
- You make no errors only if you don't program!

Basic syntax

- Names and keywords are case sensitive (camelCase).
- Remember to define variables before use.
- Variables have data types: byte, char, int, long, float…
- A line terminates with a semicolon (;).
- Enclose a block of code or function with curly braces {}
- Call a function with name(parameters) Eg: delay(1000);
- Indent your code for better clarity.
- An object can be used by calling its methods such as: Object.Method(parameter) Eg: Serial.print("Hello");

Arduino IDE

- Type your code in the window and press upload
- Includes many libraries to drive hardware
- Many 3rd party open source libraries to choose from

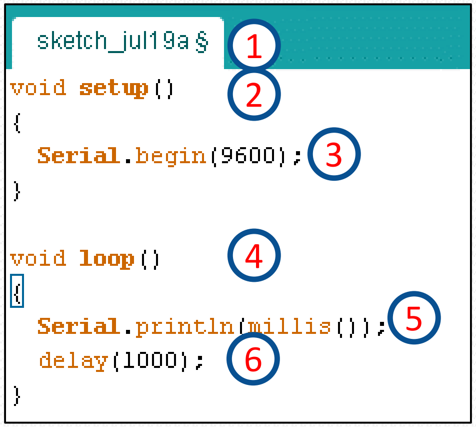

File Edit Sketch Tools Help
sketch_jul19a
1
2
3
4
5
6
Arduino M 7 2560 or Mega ADK on 8 5

1. Compile (check for error)
2. Compile and upload
3. Open serial port monitor
4. Code (sketch) area
5. Error report and information
6. Line number
7. Board (see below)
8. Serial port (see below)

Tools->Board-> “Open source physics lab V 2.4.x”
Tools->Port choose the correct port.

Simple Arduino code

- All keywords and names are CASE SENSITIVE!
- setup() runs once right after the board restarts so put initialization code in it.
- loop() runs repeatedly once setup() finishes.



```
sketch_jul19a §
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println(millis());
  delay(1000);
}
```

1. File name (§ means not yet saved)
2. setup() is a function that takes no arguments (empty parentheses) and returns nothing (void)
3. Code in a function is enclosed by curly braces ({}) and each line ends with a semicolon (;).
4. loop() is another function with no arguments or return values. Your main routine should reside here.
5. Serial.println() sends texts to PC serial monitor
6. All library functions and keywords are in orange color.

Reading voltage

Reading Arduino analog input:

```
Serial.println(analogRead(A0));
```

1024 integer values

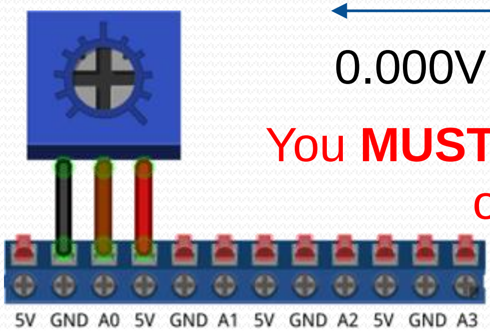
0 1023

Converting into voltage in volts

```
Serial.println(analogRead(A0)*5.0/1024);
```

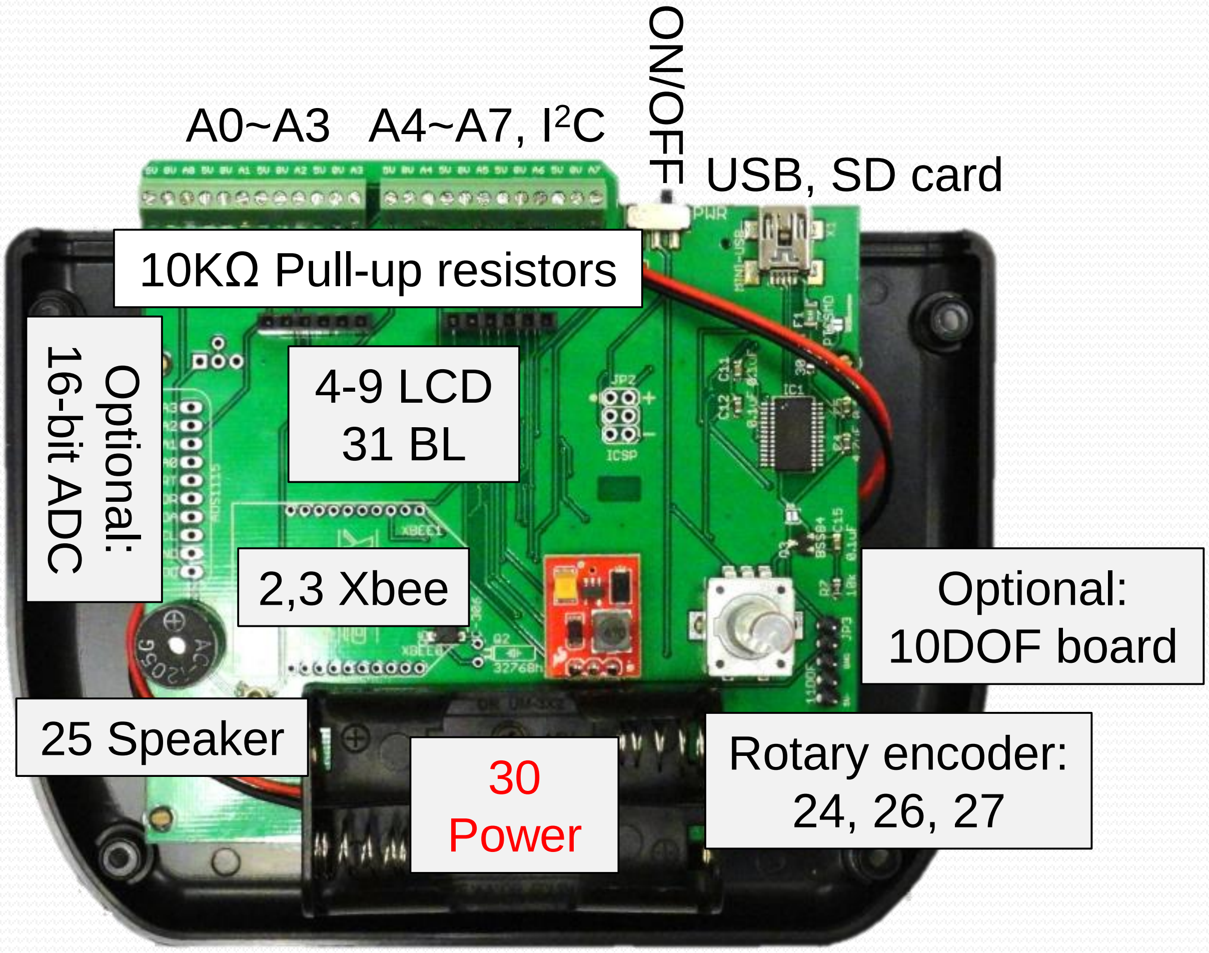
1024 floating point voltage values

0.000V 4.995V



You **MUST** use at least one floating point number such as 5.0
otherwise Arduino WILL use integer math!
5/1024=0 in integer math

OSPL Developer’s Edition



A0~A3

A4~A7, I^{2}C

ON/OFF

USB, SD card

10KΩ Pull-up resistors

Optional:
16-bit ADC

4-9 LCD
31 BL

2,3 Xbee

Optional:
10DOF board

25 Speaker

30
Power

Rotary encoder:
24, 26, 27

OSPL Developer’s Edition

- Analog pins are A0, A1, … A7
- Left screw terminal block has A0-A3, 5V, and GND
- Right screw terminal block has A4-A7, 5V, and GND
- Any of the A0-A7 can have 10Kohm pull-up via jumper
- A6 and A7 can be connected to I^2C via solder jumpers

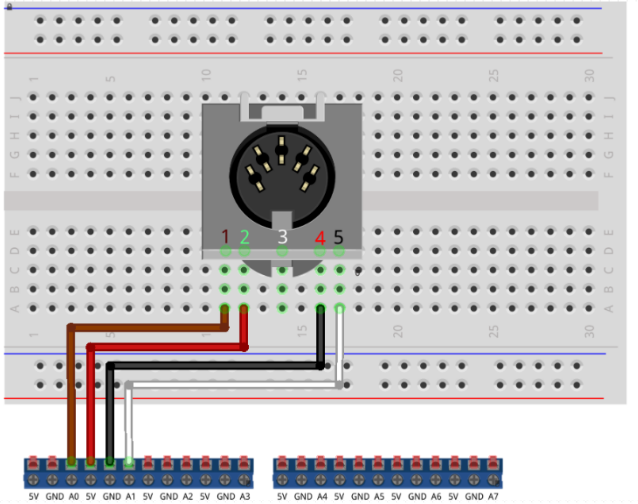
Pins	Function	Pins	Function	Pins	Function
0, 1	Serial port	14-21	Same as A0-A7	28-29	Unused
2, 3	Second serial port	22,23	I2C bus	30	Power control
4-9	LCD	24, 26, 27	Rotary encoder	31	LCD back light
10-13	micro SD card	25	Speaker		

Analog sensors

- Outputs a voltage linear to the measurement
- Measurement = Slope * Voltage + Intercept
- Vernier direct temperature probe (DCT-DIN)
- Slope 55.55DegC/V or 100DegF/V
- Intercept -17.7DegC or 0DegF
- Analog Device ADXL335 accelerometer
- Slope 32.67m/s^2/V or 300mV/g at 3V supply voltage
- Reading is ratiometric, meaning output proportional to supply voltage so with 3.3V supply it is 330mV/g.

Wiring a Vernier analog sensor

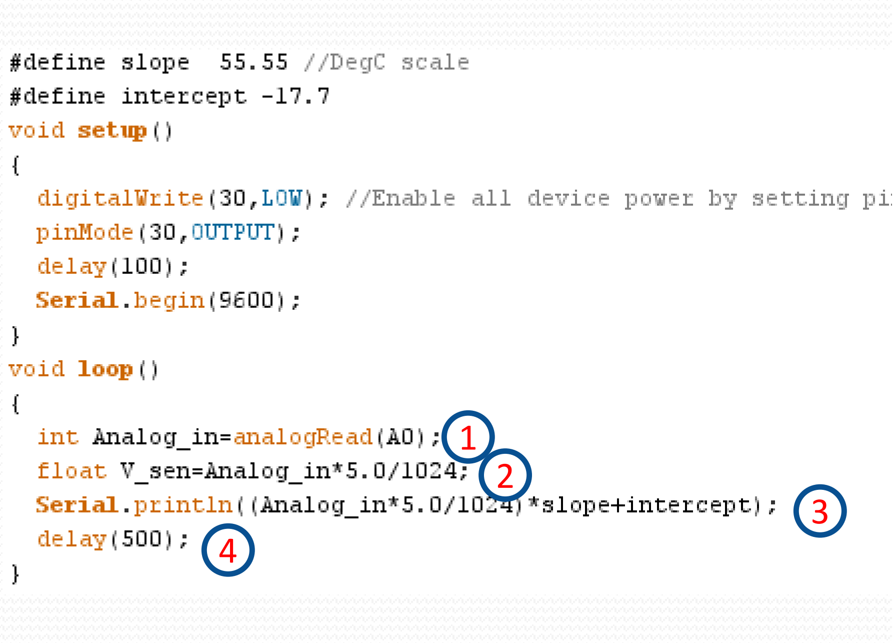
- Plug a DIN-5 sensor into a DIN-5 socket on an OSPL standard edition
- Use a BTA to DIN-5 adapter for newer Vernier sensors with BTA connectors
- For developer's edition, use the following diagram:


1
2
3
4
5
5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

Dev. edition: Use any analog pin **without** 10Kohm pull-up resistor (A0, A2, A4, A6)
Wiring: socket facing right first pin in D11
Pin1: A0 (or any of A0, A2, A4, A6)
Pin2: 5V
Pin4: GND
Optional pin5: A1(or any of A1, A3, A5, A7)

Vernier Direct Temperature

A simple program usually involves initializing the hardware, and then sensing raw data and converting raw data into suitable format for display or logging. This sensor is nothing more than an LM34CH sensor packaged in a probe with a DIN-5 connector for convenience.



```
#define slope  55.55 //DegC scale
#define intercept -17.7
void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pi
  pinMode(30,OUTPUT);
  delay(100);
  Serial.begin(9600);
}
void loop()
{
  int Analog_in=analogRead(A0);
  float V_sen=Analog_in*5.0/1024;
  Serial.println((Analog_in*5.0/1024)*slope+intercept);
  delay(500);
}
```

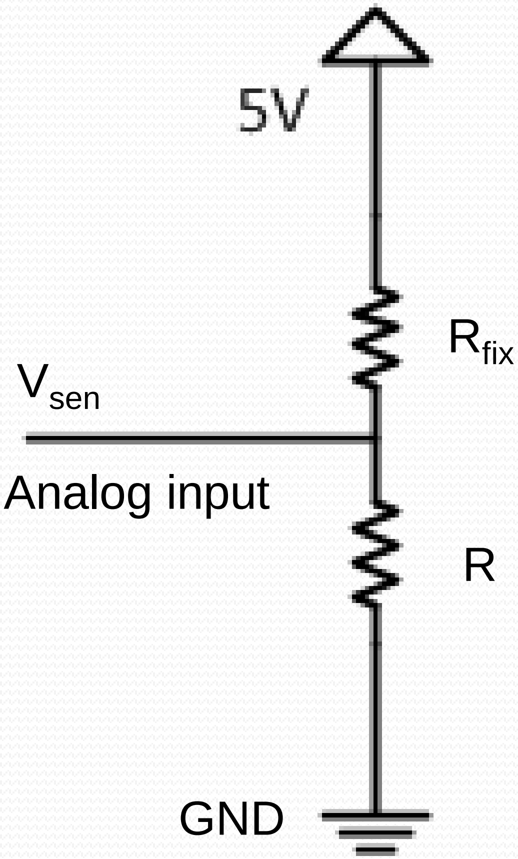
1. Acquire analog reading at channel A0. The return value is between 0 and 1023, representing 0V-5V range.
2. Output useful information for reading.
3. Convert the raw voltage into temperature in Celsius.
4. Maintain a 2 points/sec data rate.

Notice the 5.0 instead of 5 forces Analog_in from integer to floating point number to maintain accuracy.

Some slopes and intercepts

Name	Slope	Intercept
Vernier direct temperature	55.55 °C/V	-17.7 °C
Vernier force gauge (±10N scale)	-4.9 N/V	12.25 N
Vernier low-g accelerometer	22.924 m/s^2/V	-51.751 m/s^2
Vernier pressure sensor	2.203 ATM/V	0.000 ATM
Vernier hand dynamometer	175.416 N/V	-19.295 N
Vernier light sensor (6000 lux scale)	1692 lux/V	0 lux
Vernier magnetic sensor (10x scale)	32.35 mT/V	-80.625 mT
TMP36 temperature sensor	100 °C/V	-50 °C

Sensing resistance



$$V_{sen} = \frac{R}{R + R_{fix}} * 5V \qquad R = R_{fix}\frac{V_{sen}}{(5V - V_{sen})}$$

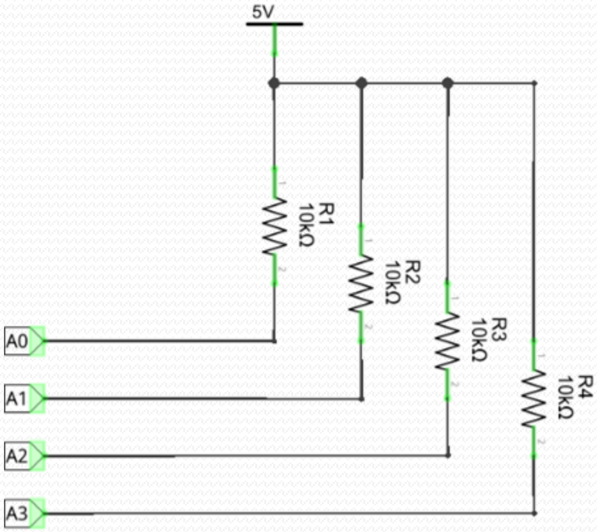
We form a voltage divider using a fixed (10Kohm) resistor and the variable resistor.

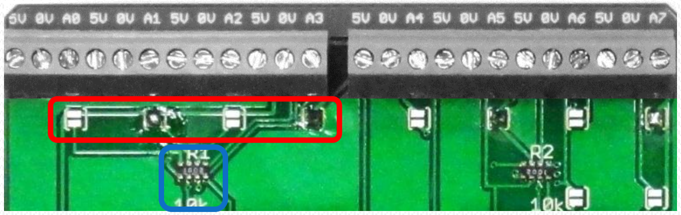
For best accuracy, the value of the fixed resistor should be equal to that of the variable resistor when it is in the middle of its range.

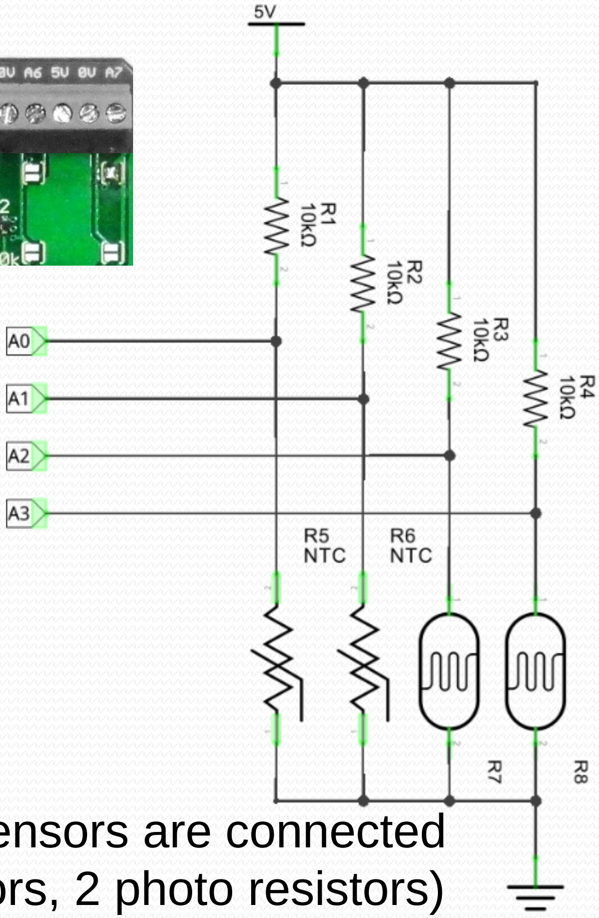
OSPL developer’s edition has options for pull-ups on all analog channels A0-A7. This makes prototyping and cabling very easy.

OSPL standard edition has on-board 10Kohm pull-up resistors on channels A1, A3, and A5 so there is no need to add a resistor to the circuit.

Pull-up resistors




5V 0V A0 5V 0V A1 5V 0V A2 5V 0V A3
5V 0V A4 5V 0V A5 5V 0V A6 5V 0V A7
R1
R2
10k

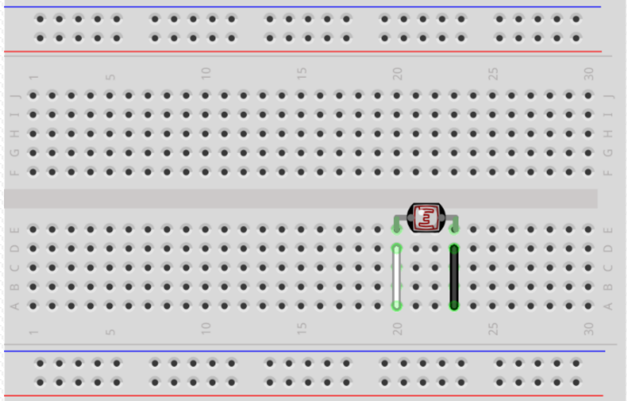


OSPL V2.4 with 10KΩ pull-up resistors
(enabled with jumpers)

Resistive sensors are connected
(2 thermistors, 2 photo resistors)

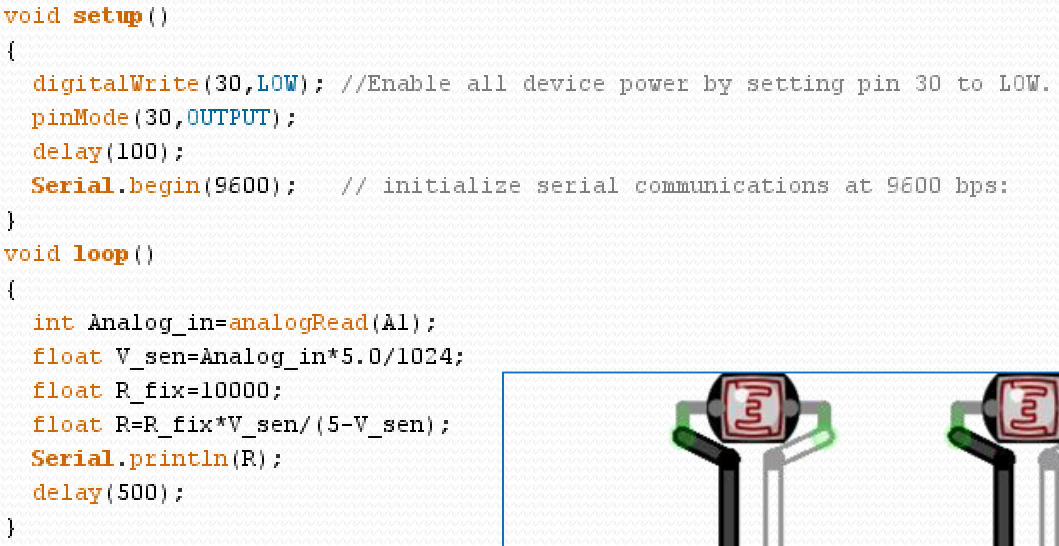
Photo resistor

$$R = R_{fix} \frac{V_{sen}}{(5V - V_{sen})}$$

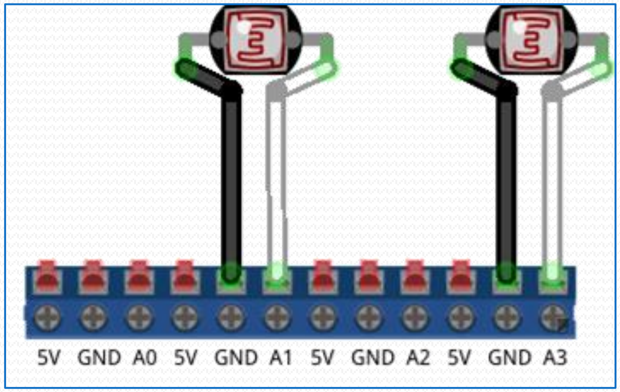


Standard edition: Use channel 0 (leftmost)
Wiring:
One end – White
Other end – Black

Integers calculation will truncate accuracy of the result. Make sure to use floating point numbers when necessary to maintain accuracy.



```
void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pin 30 to LOW.
  pinMode(30,OUTPUT);
  delay(100);
  Serial.begin(9600);    // initialize serial communications at 9600 bps:
}
void loop()
{
  int Analog_in=analogRead(A1);
  float V_sen=Analog_in*5.0/1024;
  float R_fix=10000;
  float R=R_fix*V_sen/(5-V_sen);
  Serial.println(R);
  delay(500);
}
```



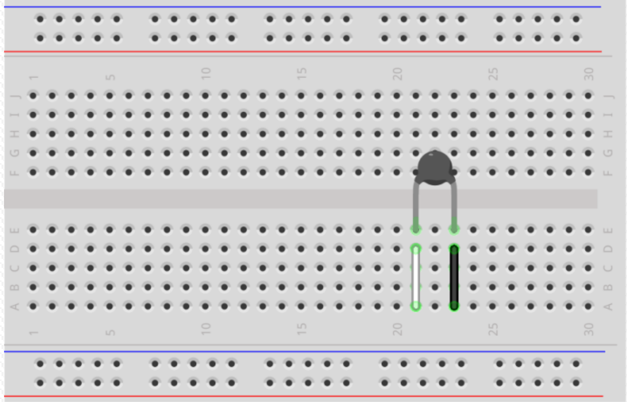
Dev. edition: Use any analog pin with 10Kohm pull-up
Wiring: one end – GND, other end – A1, A3, A5, or A7

Thermistor:

$$R = R_{fix}\frac{V_{sen}}{(5V - V_{sen})}$$

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right)$$

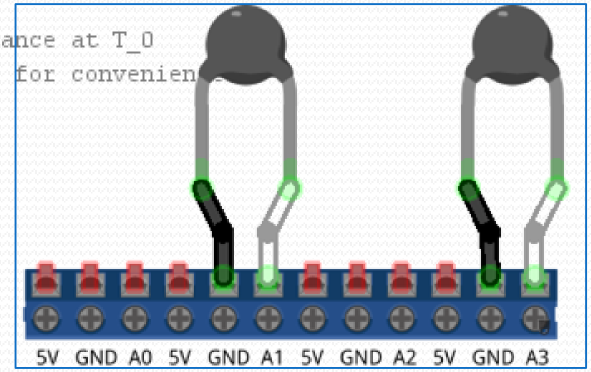
$$T = \frac{B}{\ln\left(\frac{R}{R_0 e^{-\frac{B}{T_0}}}\right)}$$



Standard edition: Use channel 0 (leftmost)
Wiring:
One end – White
Other end – Black

Integers calculation will truncate accuracy of the result. Make sure to use floating point numbers.

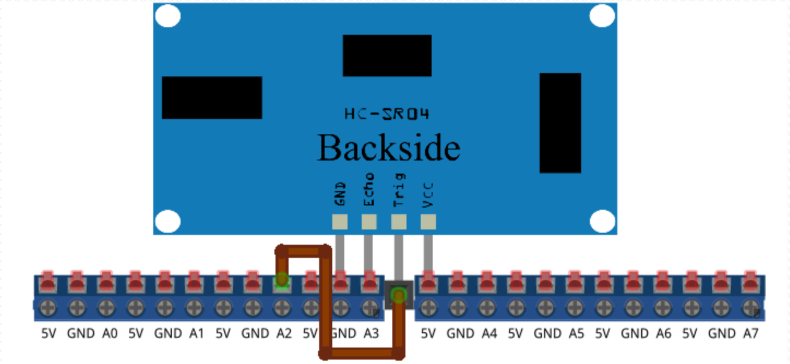
```
void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pin 30 to LOW.
  pinMode(30,OUTPUT);
  delay(100);
  Serial.begin(9600);   // initialize serial communications at 9600 bps:
}
void loop()
{
  int Analog_in=analogRead(A1);
  float V_sen=Analog_in*5.0/1024;
  float R_fix=10000;
  float R=R_fix*V_sen/(5-V_sen);
  float B=3435;//3435K for our thermistor
  float T_0=298;//298K as T_0
  float R_0=10000;//Thermistor resistance at T_0
  float r_inf=R_0*exp(-B/T_0);//R_inf for convenien
  float T=B/log(R/r_inf);
  Serial.println(T-273);//Deg C
  delay(500);
}
```



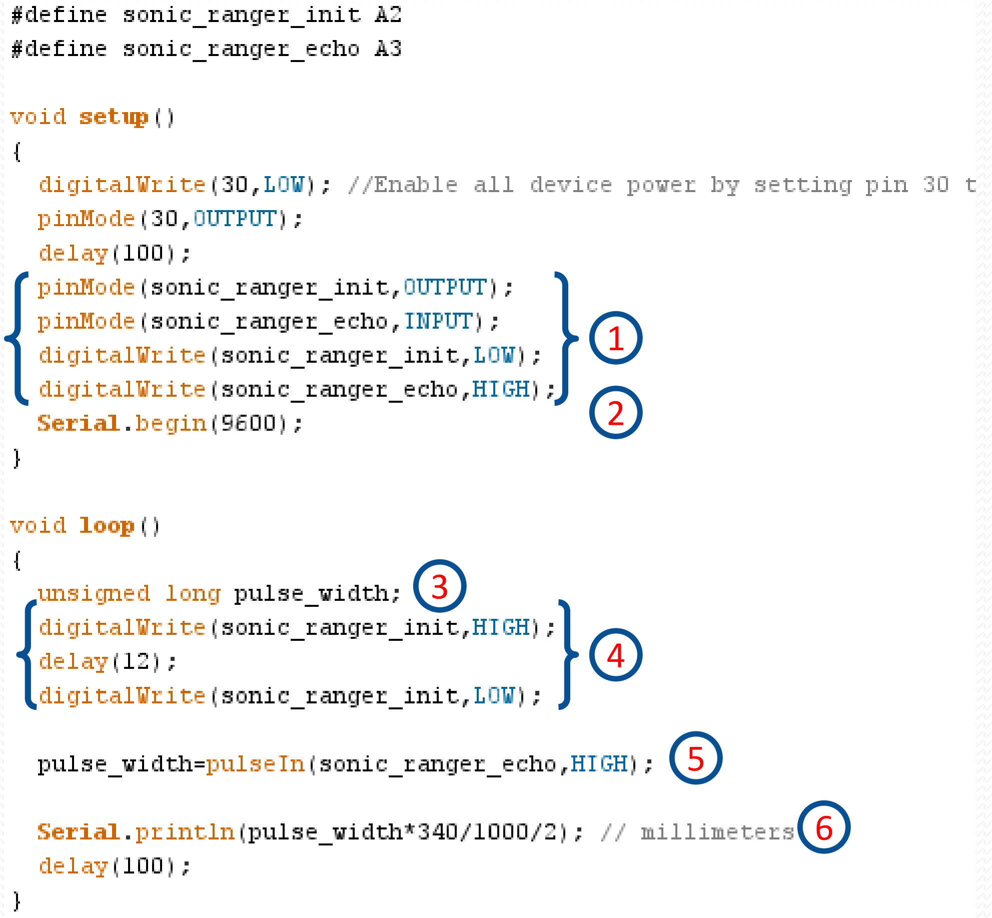
Dev. edition: Use any analog pin with 10Kohm pull-up
Wiring: one end – GND, other end – A1, A3, A5, or A7

Sonic ranger code and wiring

Connection diagram:


HC-SR04
Backside
GND
Echo
Trig
VCC
5V GND A0 5V GND A1 5V GND A2 5V GND A3
5V GND A4 5V GND A5 5V GND A6 5V GND A7

Connect a male-female wire (brown) between “Trig” and A2.

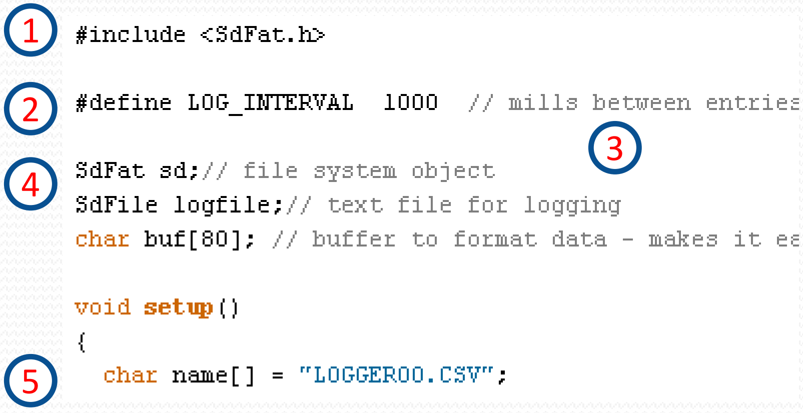


```
#define sonic_ranger_init A2
#define sonic_ranger_echo A3

void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pin 30 t
  pinMode(30,OUTPUT);
  delay(100);
  pinMode(sonic_ranger_init,OUTPUT);
  pinMode(sonic_ranger_echo,INPUT);
  digitalWrite(sonic_ranger_init,LOW);
  digitalWrite(sonic_ranger_echo,HIGH);
  Serial.begin(9600);
}

void loop()
{
  unsigned long pulse_width;
  digitalWrite(sonic_ranger_init,HIGH);
  delay(12);
  digitalWrite(sonic_ranger_init,LOW);

  pulse_width=pulseIn(sonic_ranger_echo,HIGH);

  Serial.println(pulse_width*340/1000/2); // millimeters
  delay(100);
}
```

1. Initialize pins A2 as output and A3 as input.
2. Initialize the serial port to 9600 baud rate.
3. Define a variable to store result.
4. Toggle the INIT pin to start ranging.
5. Get result with a function pulseIn().
6. Convert the result into distance in mm and print it to serial monitor.

More complex code

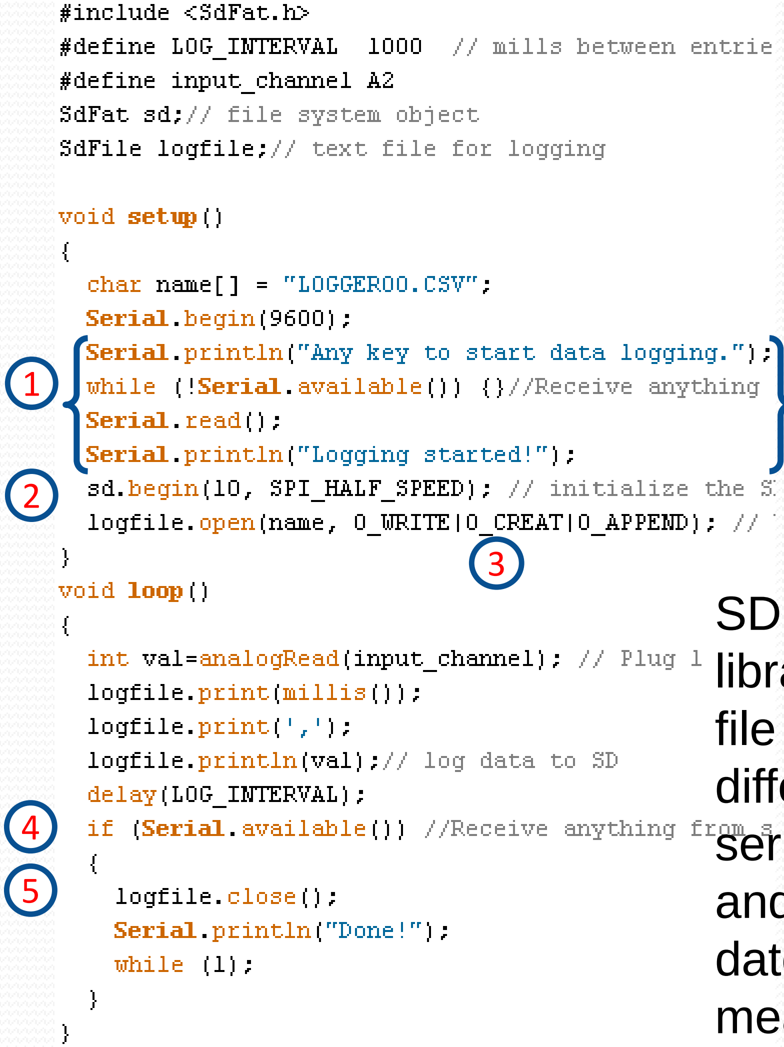
- #include <xyz.h> uses library xyz in the code
- #define abc 123 makes it easy to use a meaningful symbol in place of number or enable certain features
- Variables are defined as *type name=value*;
- Variables defined outside (inside) functions is global (local) and can be used anywhere (in the function)



```
#include <SdFat.h>

#define LOG_INTERVAL  1000  // mills between entries

SdFat sd;// file system object
SdFile logfile;// text file for logging
char buf[80]; // buffer to format data - makes it ea

void setup()
{
  char name[] = "LOGGER00.CSV";
```

1. Includes the SdFat library for SD card
2. Define LOG_INTERVAL as 1000. Use LOG_INTERVAL instead of 1000 in code.
3. Comments are after double slash (//).
4. Global variables. *sd* is SdFat type, *logfile* is SdFile type, *buf* is char array type.
5. *name* is a local variable only in setup.

SD logging



```
#include <SdFat.h>
#define LOG_INTERVAL  1000  // mills between entrie
#define input_channel A2
SdFat sd;// file system object
SdFile logfile;// text file for logging

void setup()
{
  char name[] = "LOGGER00.CSV";
  Serial.begin(9600);
  Serial.println("Any key to start data logging.");
  while (!Serial.available()) {}//Receive anything
  Serial.read();
  Serial.println("Logging started!");
  sd.begin(10, SPI_HALF_SPEED); // initialize the S
  logfile.open(name, O_WRITE|O_CREAT|O_APPEND); //
}
void loop()
{
  int val=analogRead(input_channel); // Plug 1
  logfile.print(millis());
  logfile.print(',');
  logfile.println(val);// log data to SD
  delay(LOG_INTERVAL);
  if (Serial.available()) //Receive anything from s
  {
    logfile.close();
    Serial.println("Done!");
    while (1);
  }
}
```

1. Wait for user input before start logging.
2. Initialize the SD card. Pin 10 is the Chip-select pin on the OSPL and some SD shields.
3. Use logfile to open LOGGER00.CSV to write to, create it if it doesn't exist, append if it does.
4. Wait for user input again before stop logging.
5. The open file must be closed to prevent data loss. One shouldn't just turn off OSPL when it is logging to SD card.

SD logging is made extremely easy with the SdFat library written by William Greiman. You can print to a file just like you can to the serial port. The only difference is that it has a lot more functions than a serial port, such as creating/renaming/deleting files and folders, testing the existence of files and adding date/time call-back functions so the file has meaningful date/time instead of 2000/1/1 12:00am.

CH4: OSPL discussion

- Device features
- A few applications
- Device cost
- Building devices
- Discussion and opportunities

OSPL Dev. Edition features

- Firmware supports lots of sensors and data logging
- Screw terminal blocks with 8 analog/digital pins
- 10Kohm pull-up resistors via solder jumpers, making wiring resistive and open-drain sensors very easy
- Micro-SD card and real-time clock for data logging
- LCD and rotary encoder for easy user interface
- Xbee socket for wireless control or data transfer
- Ability to turn off all peripherals to conserve battery

A few applications

- Digital self-calibrating hydrometer using force gauge and servo motor
- Center of mass visualizer using 3 force gauges, a web camera and a laptop
- Sample lab for finding relation between altitude and pressure using the 10 DOF board
- Smart track using 3 force gauges that displays location of a cart on a track and simulates sonic ranger to run on Vernier interface.

Application: digital hydrometer

- Applies introductory physics in liquid density sensing
- Automatically recalibrates at startup
- Continuously reports density of liquid
- Sending data wirelessly via Bluetooth
- Potential to add Text-to-speech for visually impaired student
- Accuracy is 0.01g/cm^3

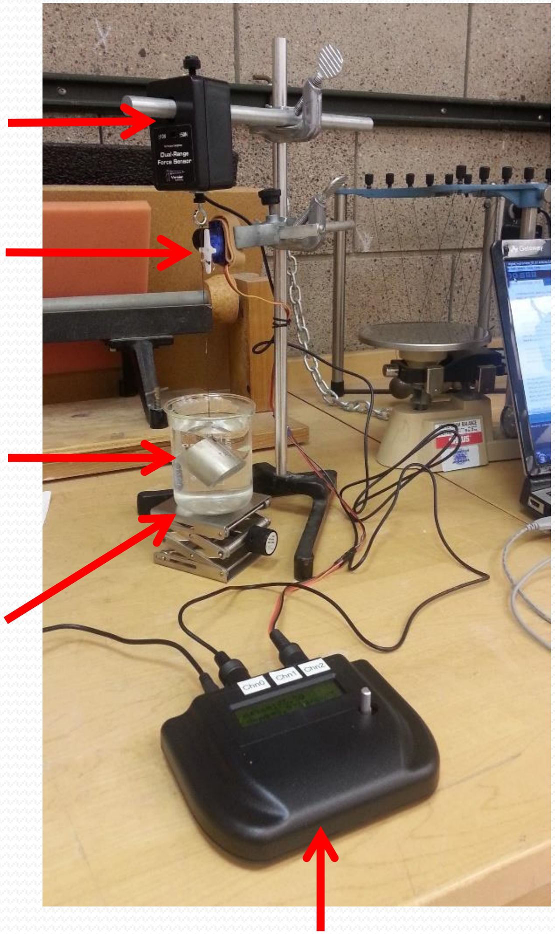
Digital hydrometer (2014)

Vernier force gauge

Servo motor

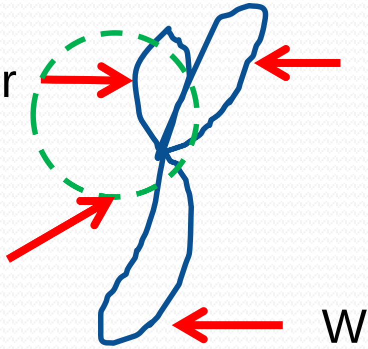
200g weight on string with loop

Liquid in beaker



OSPL V2.0

Wire loop for weighing and auto calibration



Motor

Force gauge

Hook motion path

Weight

Density reading

Application: COM visualizer

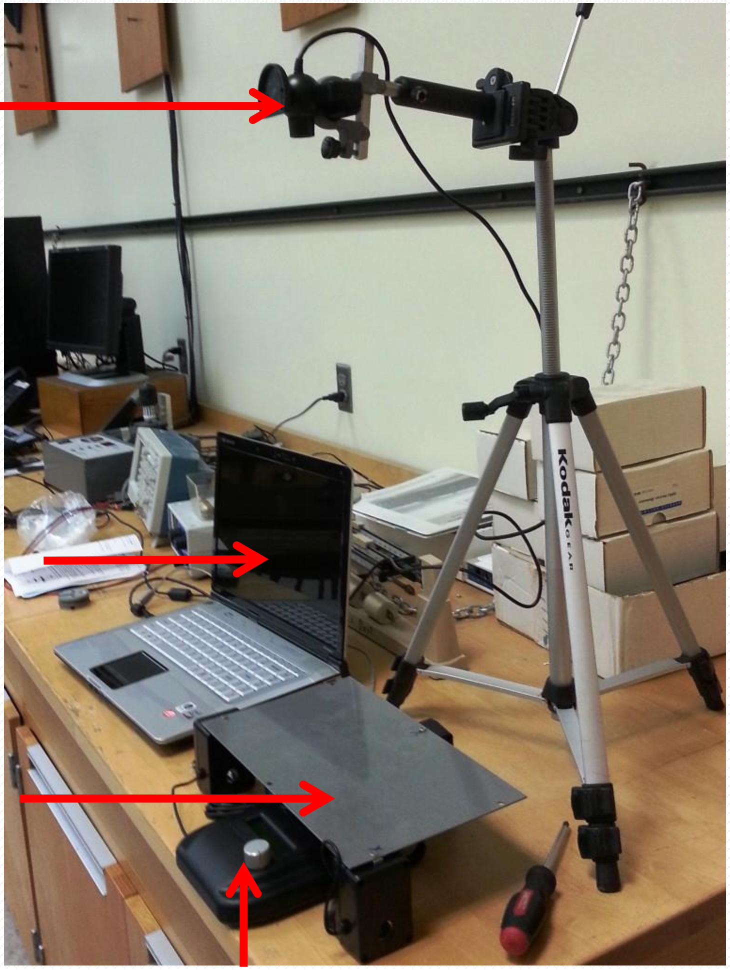
- Applies introductory physics in finding center of mass
- Automatically finds the center of mass by weighing
- Continuously reports COM location to PC
- PC uses webcam and video calibration to show COM in LIVE video
- Very nice demo for intro physics!

COM visualizer (2013)

Webcam

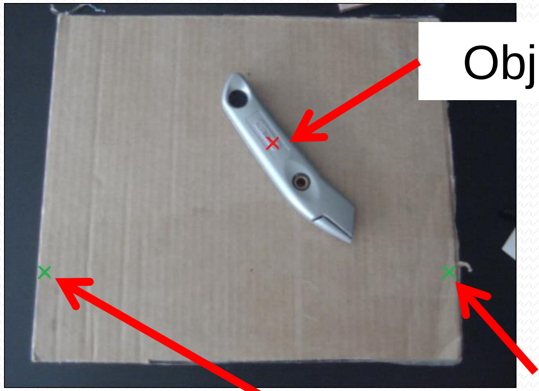
Computer with live video of COM

Weighing plate with 3 Vernier force gauges



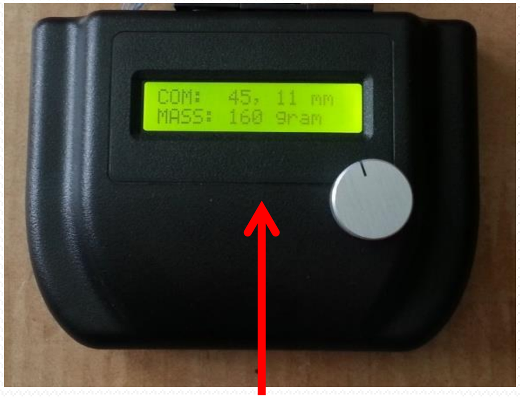
OSPL V2.0

COM in LIVE video (red cross)



Object

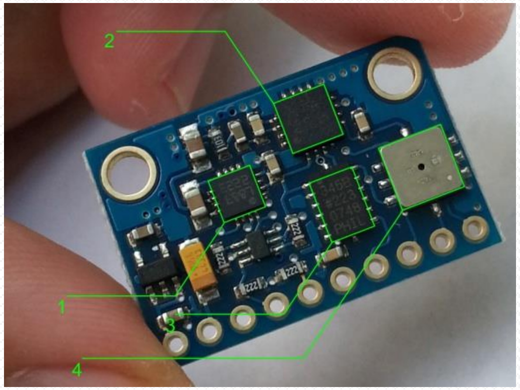
Calibration markers



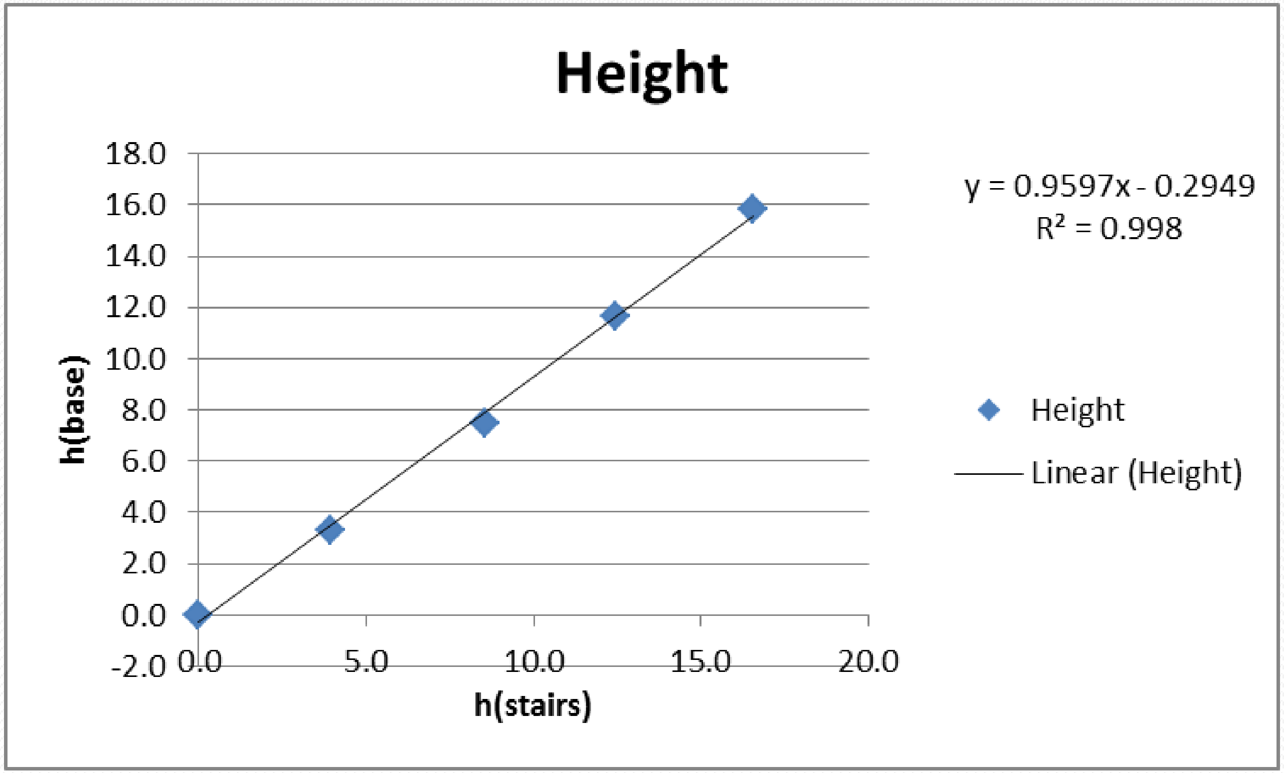
Local reading

Application: altitude vs. pressure

- Assumption: altitude is linearly related to pressure at low altitude and when altitude change is small
- Students can measure the height of a building from the number of stairs and height of each stair
- Students can also measure pressure at each floor with OSSL and a pressure gauge (included in the $13 board)
- They calculate height from pressure
- They confirm their assumption



Altitude vs. pressure



$$P = P_b \cdot \exp\left[\frac{-g_0 \cdot M \cdot (h - h_b)}{R^* \cdot T_b}\right]$$

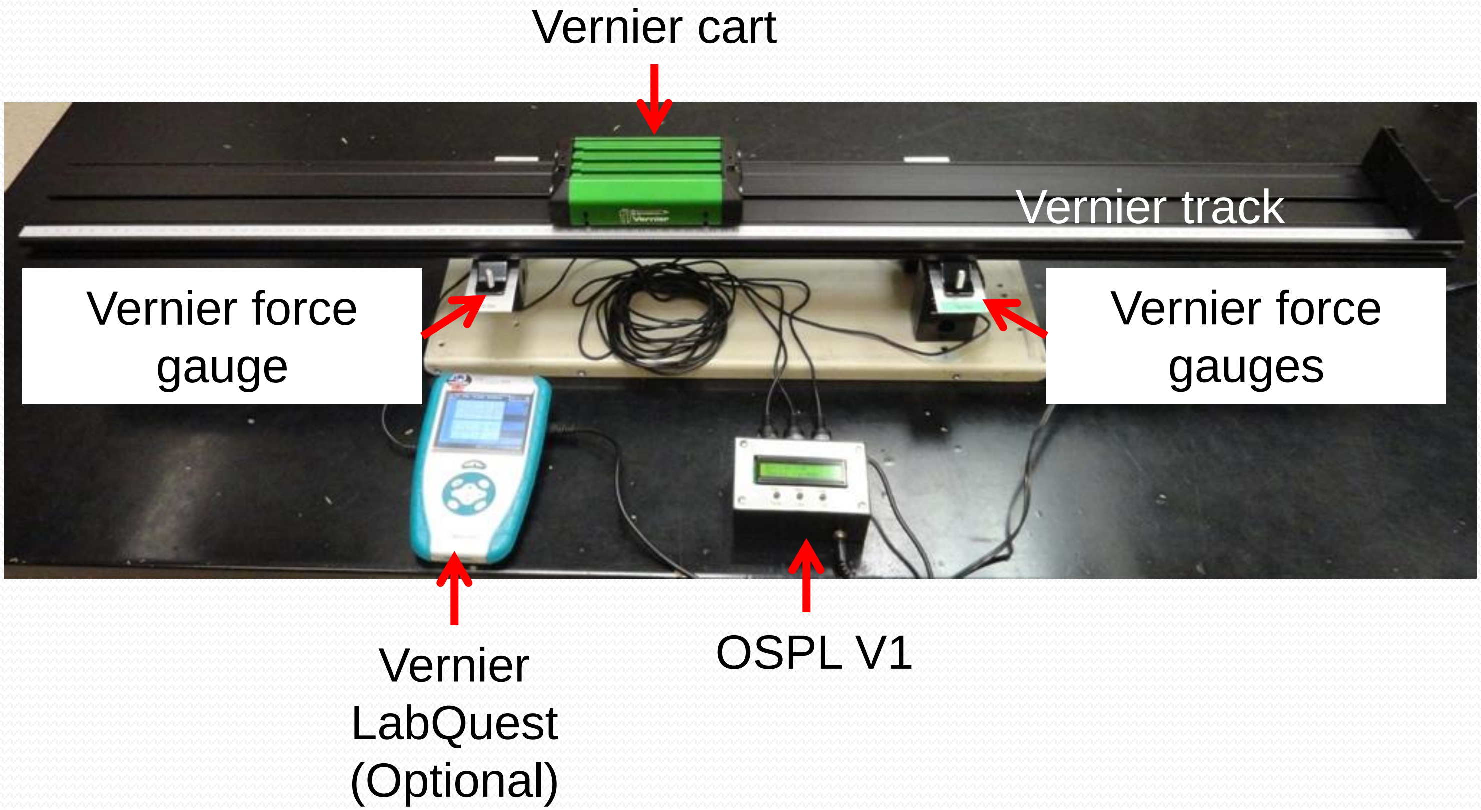
From wikipedia.org

Obtained at Wick Science Building between the basement and the 4th floor.

Application: smart track

- Automatically reports location of a cart on a track
- Emulates a sonic ranger when connected to Vernier LabQuest to log position velocity and acceleration
- Running averages of positions
- Tare function
- See how sonic rangers work with Vernier LabQuest
- See how physics applies to real life situations

Smart track (2012)

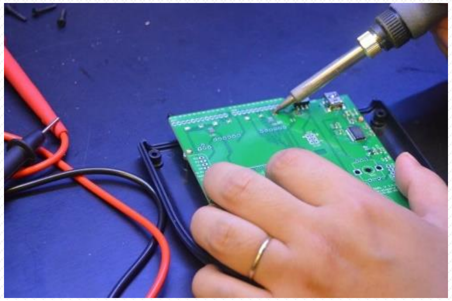


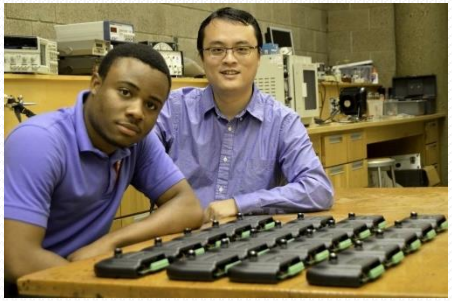
OSPL Cost:

- $65 for parts (1 unit) for dev. edition
- $55 for parts (25 unit) for dev. edition
- Standalone and needs no PC or mobile device
- Sensors can be constructed from parts for a few dollars
- Each unit build and troubleshooting takes 1 hour
- Akan and Dr. Liu spent weeks to plan and execute the build, including building tools for assembly and tests
- We also went over the budget ($80/person) a bit

This build

1120 solder junctions
720 surface mount components
240 screws





Future builds

- Will go through assembly and testing service
- Will be able to build batches of 20 to 100 units within reasonable time frames
- Need time to research potential assembly service providers and pricing (domestic vs. overseas, quantity)
- Need some seed money to kick start the process
- Writing grant proposal, willing to join effort
- Reach out to the data logger world and possibly start a kickstarter.com project

Discussions/Opportunities:

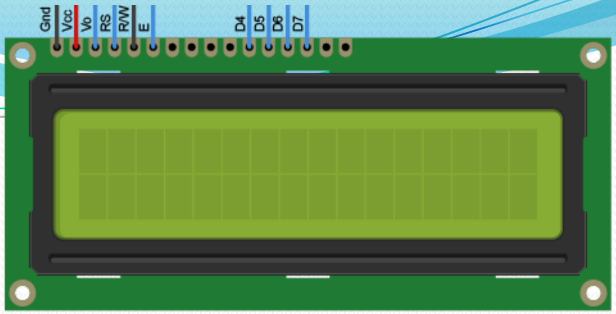
- Apply for grants
- Pilot the OSPL in your institution
- Build course materials and/or apparatus with OSPL
- Bring OSPL to high schools and middle schools

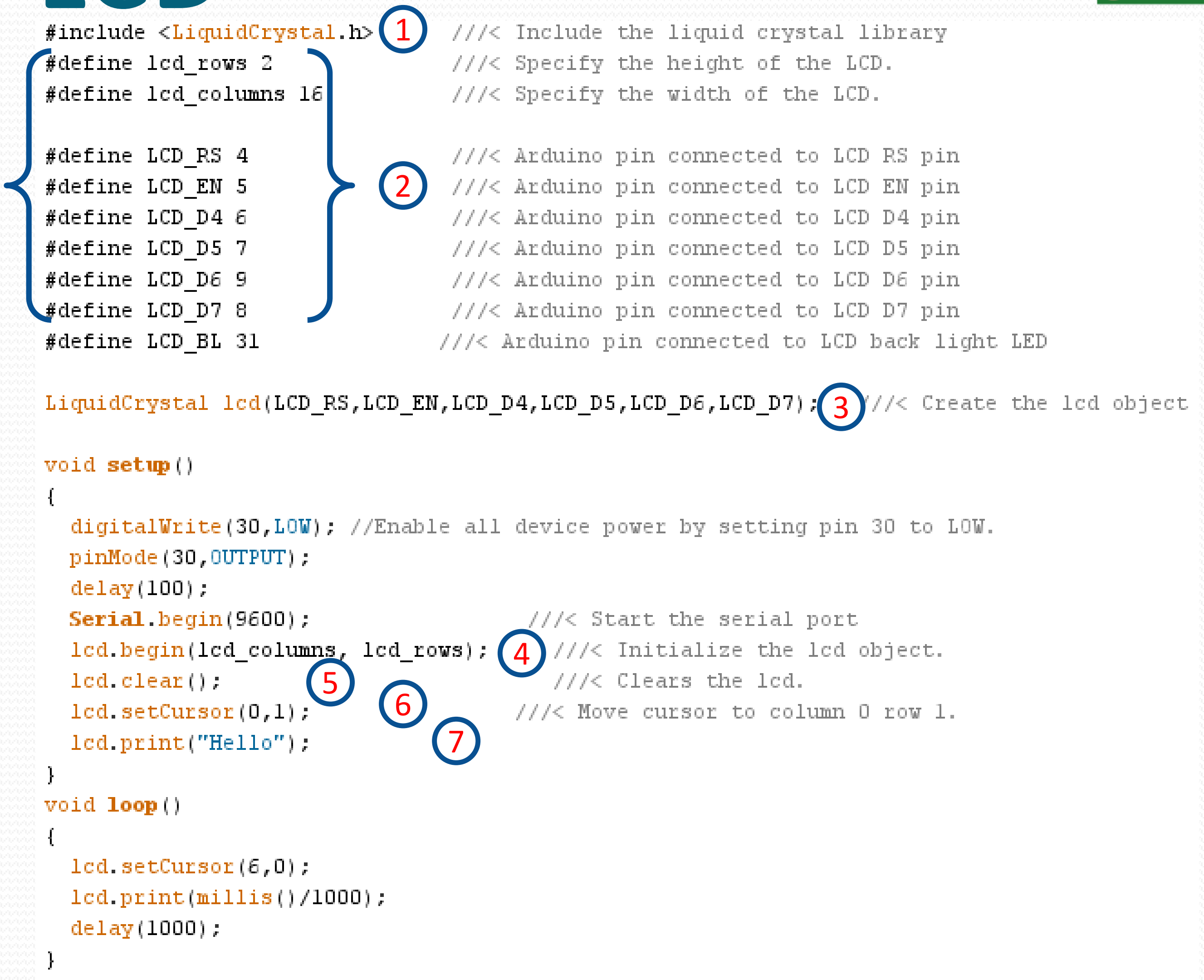
- More learning and working needed for instructors
- Many sensors don’t have proper mounting options
- Need enough users to sustain a community of support
- Low profit margin for many for-profit apparatus vendors to get involved

CH5: Additional OSPL features

- How to use the LCD
- How to use the rotary encoder
- Simple user interface
- How to use the micro-SD card
- How to use the real-time clock
- How to use the Xbee socket

LCD





```
#include <LiquidCrystal.h>          ///< Include the liquid crystal library
#define lcd_rows 2                  ///< Specify the height of the LCD.
#define lcd_columns 16              ///< Specify the width of the LCD.

#define LCD_RS 4                    ///< Arduino pin connected to LCD RS pin
#define LCD_EN 5                    ///< Arduino pin connected to LCD EN pin
#define LCD_D4 6                    ///< Arduino pin connected to LCD D4 pin
#define LCD_D5 7                    ///< Arduino pin connected to LCD D5 pin
#define LCD_D6 9                    ///< Arduino pin connected to LCD D6 pin
#define LCD_D7 8                    ///< Arduino pin connected to LCD D7 pin
#define LCD_BL 31                  ///< Arduino pin connected to LCD back light LED

LiquidCrystal lcd(LCD_RS,LCD_EN,LCD_D4,LCD_D5,LCD_D6,LCD_D7); ///< Create the lcd object

void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pin 30 to LOW.
  pinMode(30,OUTPUT);
  delay(100);
  Serial.begin(9600);                ///< Start the serial port
  lcd.begin(lcd_columns, lcd_rows);  ///< Initialize the lcd object.
  lcd.clear();                        ///< Clears the lcd.
  lcd.setCursor(0,1);               ///< Move cursor to column 0 row 1.
  lcd.print("Hello");
}
void loop()
{
  lcd.setCursor(6,0);
  lcd.print(millis()/1000);
  delay(1000);
}
```

1. Include library
2. LCD size and pins used to drive it.
3. Create lcd object
4. Initialize lcd
5. Clear lcd
6. Set cursor before every print.
7. There is only print and no println.

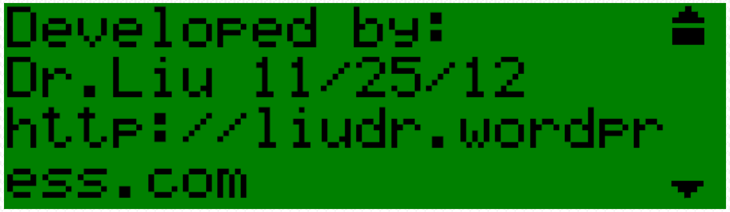
Printing on lcd takes tens of milliseconds so don’t print to it when sensing photo gates.

Rotary encoder

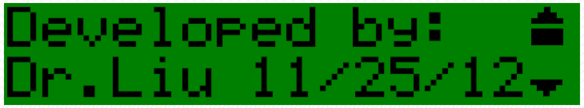
```
#include <phi_interfaces.h> 1 ///< Include the phi_interfaces input devices library

#define total_buttons 1             ///< The total number of push buttons in a buttons group object. This is needed to insta
#define encoder_detents 18          ///< How many detents per rotation
#define ch_b 27                   2 ///< I/O pin for rotary encoder channel b
#define ch_a 26                     ///< I/O pin for rotary encoder channel a
#define btn_b 24                    ///< I/O pin for a button

byte pins_buttons[]={btn_b};          ///< The digital pins connected to the buttons are included.
char mapping_buttons[]={'B'};         ///< This is a list of names for each button.
byte pins_encoder[]={ch_a,ch_b};    3 ///< The digital pins connected to rotary encoder channels.
char mapping_encoder[]={'U','D'};     ///< This is a list of names for each button.

// The following lines instantiate several keypads.                                   4
phi_button_groups my_btns(mapping_buttons, pins_buttons, total_buttons);    ///< This instantiates a button group to sen
phi_rotary_encoders rotary_keypad(mapping_encoder,pins_encoder[0],pins_encoder[1],encoder_detents);    ///< This is the

void setup()
{
  digitalWrite(30,LOW); //Enable all device power by setting pin 30 to LOW.
  pinMode(30,OUTPUT);
  delay(100);
  Serial.begin(9600);                   ///< Start the serial port for data upload and serial keypad.
}
void loop()
{
5 char btn_in=my_btns.getKey(); // User getKey() to sense the input
6 if (btn_in!=NO_KEY) Serial.println(btn_in); // NO_KEY should not be responded
  char encoder_in=rotary_keypad.getKey(); // User getKey() to sense the input
  if (encoder_in!=NO_KEY) Serial.println(encoder_in); // NO_KEY should not be responded
}
```

Messages and lists

- Display a long scrollable message on the LCD, use rotary encoder to read it and dismiss with shaft click
- simple_text_area(message_to_display);



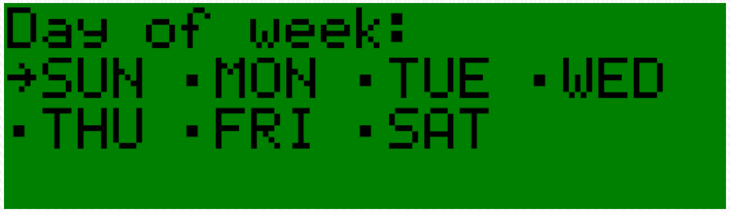
```
Developed by:      ▲
Dr.Liu 11/25/12
http://liudr.wordpr
ess.com            ▼
```



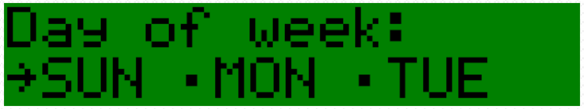
```
Developed by:   ▲
Dr.Liu 11/25/12▼
```

simple_text_area("Developed by:\nDr.Liu 11/25/12\nhttp://liudr.wordpress.com\nThis is an example.\nPress Confirm to continue");

- Display a select list with title on the LCD, use rotary encoder to highlight an item and shaft click to select
- int choice=simple_select_list(list_with_title);



```
Day of week:
→SUN ·MON ·TUE ·WED
·THU ·FRI ·SAT
```

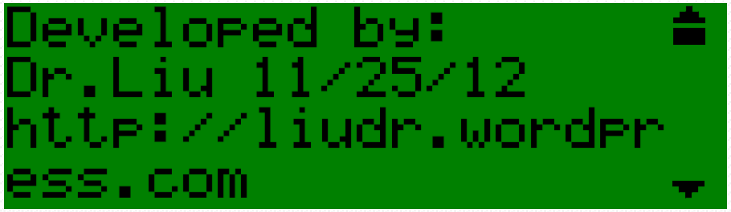


```
Day of week:
→SUN ·MON ·TUE
```

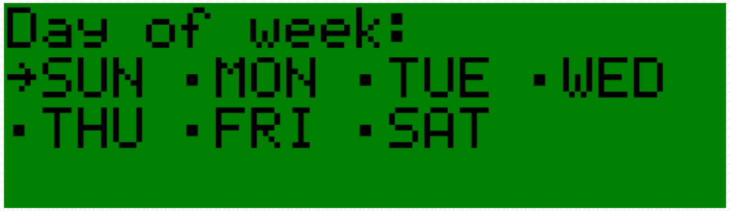
response=simple_select_list("Day of week:\nSUN\nMON\nTUE\nWED\nTHU\nFRI\nSAT\n");

Simple user interface setup

1. Include the libraries
2. Define the relevant parameters for the hardware, such as pins etc.
3. Define what button output represents what function, such as ‘B’ button means enter ‘U’ button means up.
4. Instantiate objects for different keypad types, including serial port keypad
5. Include all keypad types in an array
6. Initialize the library for use
7. Set up some simple parameters
8. Call library functions to interact with the user.
9. You can also call wait_on_escape(time_ms) to get button pushes from all input devices.
10. You may substitute or simulate button presses with serial port monitor.

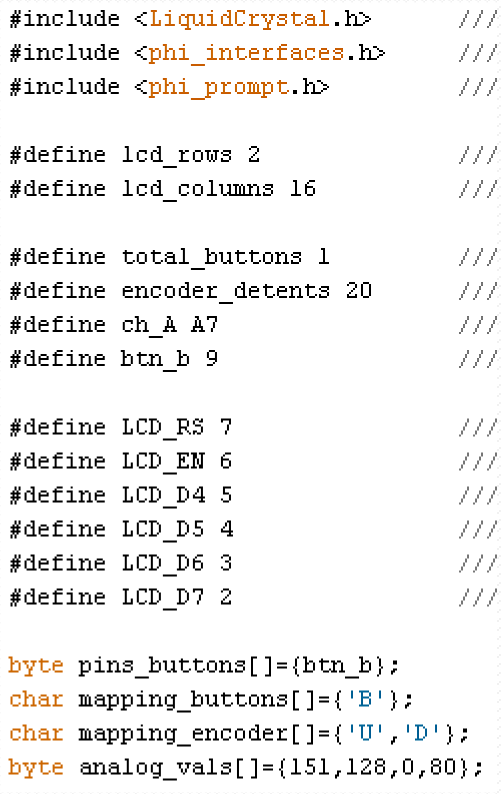


```
Developed by:       ▲
Dr.Liu 11/25/12
http://liudr.wordpr
ess.com             ▼
```

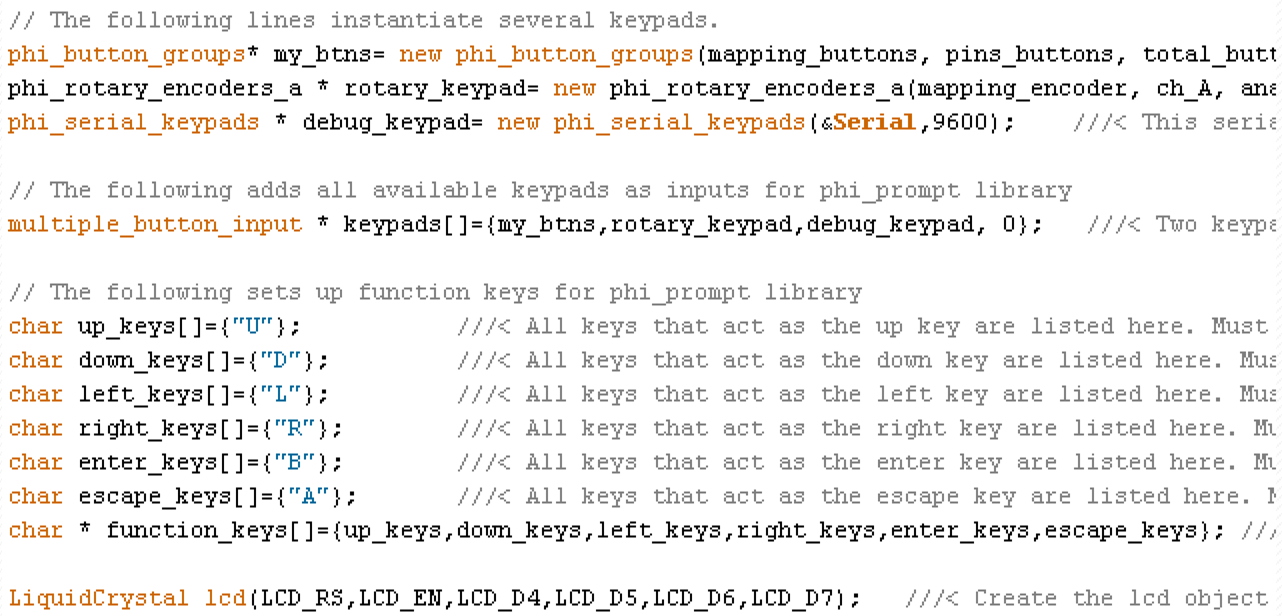


```
Day of week:
→SUN ·MON ·TUE ·WED
·THU ·FRI ·SAT
```

Initial setup (don't change)

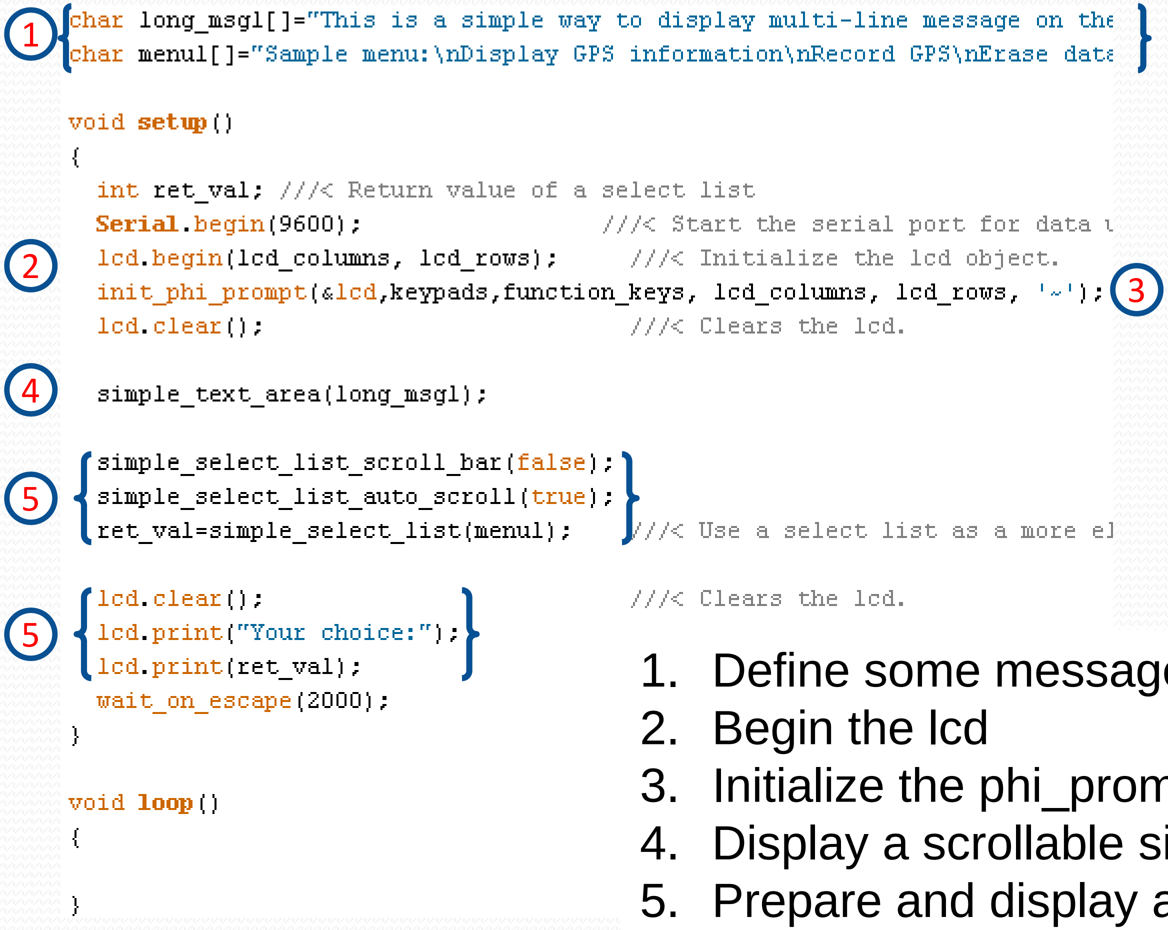


```
#include <LiquidCrystal.h>       ///
#include <phi_interfaces.h>      ///
#include <phi_prompt.h>          ///

#define lcd_rows 2               ///
#define lcd_columns 16           ///

#define total_buttons 1          ///
#define encoder_detents 20       ///
#define ch_A A7                  ///
#define btn_b 9                  ///

#define LCD_RS 7                 ///
#define LCD_EN 6                 ///
#define LCD_D4 5                 ///
#define LCD_D5 4                 ///
#define LCD_D6 3                 ///
#define LCD_D7 2                 ///

byte pins_buttons[]={btn_b};
char mapping_buttons[]={'B'};
char mapping_encoder[]={'U','D'};
byte analog_vals[]={151,128,0,80};
```



```
// The following lines instantiate several keypads.
phi_button_groups* my_btns= new phi_button_groups(mapping_buttons, pins_buttons, total_butt
phi_rotary_encoders_a * rotary_keypad= new phi_rotary_encoders_a(mapping_encoder, ch_A, ana
phi_serial_keypads * debug_keypad= new phi_serial_keypads(&Serial,9600);    ///< This seria

// The following adds all available keypads as inputs for phi_prompt library
multiple_button_input * keypads[]={my_btns,rotary_keypad,debug_keypad, 0};   ///< Two keypa

// The following sets up function keys for phi_prompt library
char up_keys[]={"U"};           ///< All keys that act as the up key are listed here. Must
char down_keys[]={"D"};         ///< All keys that act as the down key are listed here. Mus
char left_keys[]={"L"};         ///< All keys that act as the left key are listed here. Mus
char right_keys[]={"R"};        ///< All keys that act as the right key are listed here. Mu
char enter_keys[]={"B"};        ///< All keys that act as the enter key are listed here. Mu
char escape_keys[]={"A"};       ///< All keys that act as the escape key are listed here. M
char * function_keys[]={up_keys,down_keys,left_keys,right_keys,enter_keys,escape_keys}; //

LiquidCrystal lcd(LCD_RS,LCD_EN,LCD_D4,LCD_D5,LCD_D6,LCD_D7);   ///< Create the lcd object
```

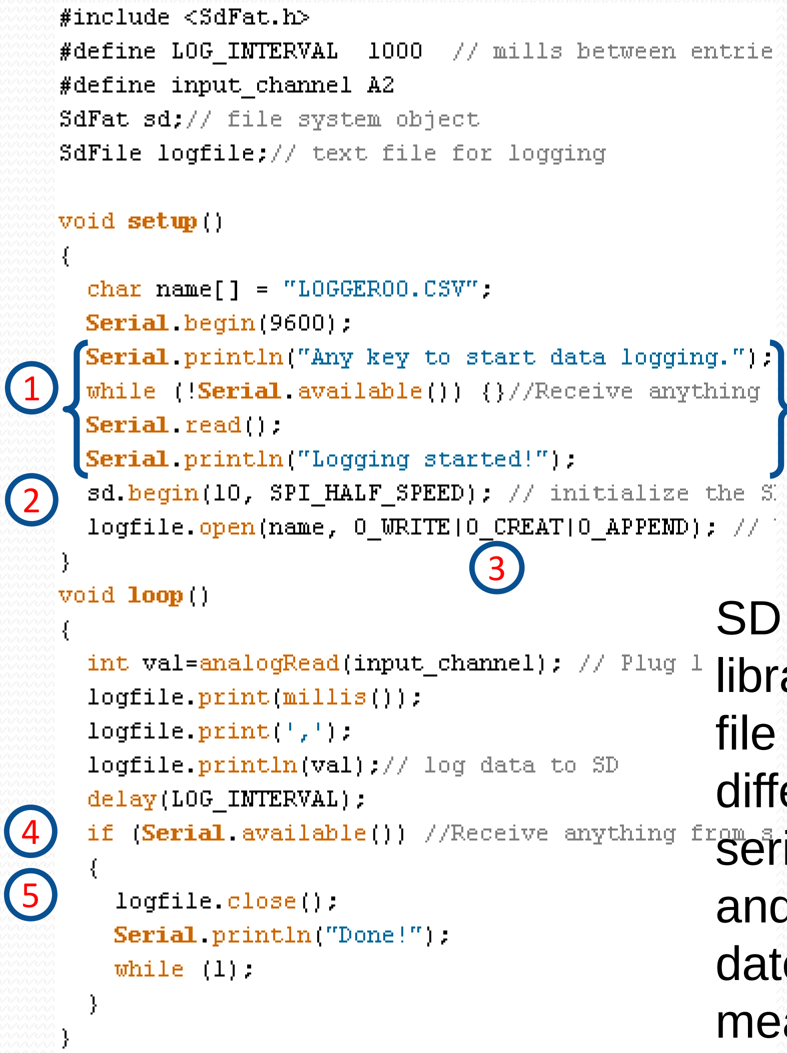
Your code



```
char long_msg1[]="This is a simple way to display multi-line message on the
char menu1[]="Sample menu:\nDisplay GPS information\nRecord GPS\nErase dat

void setup()
{
  int ret_val; ///< Return value of a select list
  Serial.begin(9600);                   ///< Start the serial port for data u
  lcd.begin(lcd_columns, lcd_rows);      ///< Initialize the lcd object.
  init_phi_prompt(&lcd,keypads,function_keys, lcd_columns, lcd_rows, '~');
  lcd.clear();                           ///< Clears the lcd.

  simple_text_area(long_msg1);

  simple_select_list_scroll_bar(false);
  simple_select_list_auto_scroll(true);
  ret_val=simple_select_list(menu1);     ///< Use a select list as a more el

  lcd.clear();                           ///< Clears the lcd.
  lcd.print("Your choice:");
  lcd.print(ret_val);
  wait_on_escape(2000);
}

void loop()
{

}
```

1. Define some messages and lists
2. Begin the lcd
3. Initialize the phi_prompt library
4. Display a scrollable simple text area
5. Prepare and display a simple select list
6. Respond to user’s choice

Micro-SD card

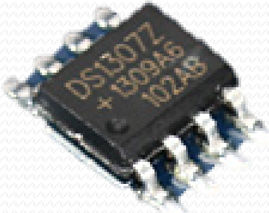


```
#include <SdFat.h>
#define LOG_INTERVAL  1000  // mills between entrie
#define input_channel A2
SdFat sd;// file system object
SdFile logfile;// text file for logging

void setup()
{
  char name[] = "LOGGER00.CSV";
  Serial.begin(9600);
  Serial.println("Any key to start data logging.");
  while (!Serial.available()) {}//Receive anything
  Serial.read();
  Serial.println("Logging started!");
  sd.begin(10, SPI_HALF_SPEED); // initialize the S
  logfile.open(name, O_WRITE|O_CREAT|O_APPEND); // `
}
void loop()
{
  int val=analogRead(input_channel); // Plug l
  logfile.print(millis());
  logfile.print(',');
  logfile.println(val);// log data to SD
  delay(LOG_INTERVAL);
  if (Serial.available()) //Receive anything from s
  {
    logfile.close();
    Serial.println("Done!");
    while (1);
  }
}
```

1. Wait for user input before start logging.
2. Initialize the SD card. Pin 10 is the Chip-select pin on the OSPL and some SD shields.
3. Use logfile to open LOGGER00.CSV to write to, create it if it doesn’t exist, append if it does.
4. Wait for user input again before stop logging.
5. The open file must be closed to prevent data loss. One shouldn’t just turn off OSPL when it is logging to SD card.

SD logging is made extremely easy with the SdFat library written by William Greiman. You can print to a file just like you can to the serial port. The only difference is that it has a lot more functions than a serial port, such as creating/renaming/deleting files and folders, testing the existence of files and adding date/time call-back functions so the file has meaningful date/time instead of 2000/1/1 12:00am.

Real-time clock



- Log data with date and time
- Save files with correct “create” and “modify” time
- Clock-driven data logging cycles
- Authenticate student data against cheating
- A good coin battery can keep time for years
- Alternative is 2000/1/1 12:00AM for everything!
- Sample code is provided

Xbee socket

- Use Serial1 to communicate with the Xbee module
- Bluetooth bees connect to phones and tablets
- Wifi xbees connect to home wireless routers
- Zigbees connect to one another to form a local mesh network
- Other xbee footprint devices include GPS and other RF modules
- Easily make internet of things (IoT)

PLEASE RETURN

- Evaluation form
- Vernier sensors
- Screw drivers
- Rulers
- Pulleys
- Magnets
- Soldering station and supplies
- Coin battery if you are concerned with airport security

Legal terms

- Circuit board design is released under Creative Commons Attribution Share-alike
- Arduino libraries are released under GNU Lesser General Public License
- OSPL firmware is released under GNU General Public License
- Contributed libraries have their own licenses

GPL 3 Free Software
Free as in Freedom

cc BY SA

LGPL 3 Free Software
Free as in Freedom

Thank you!