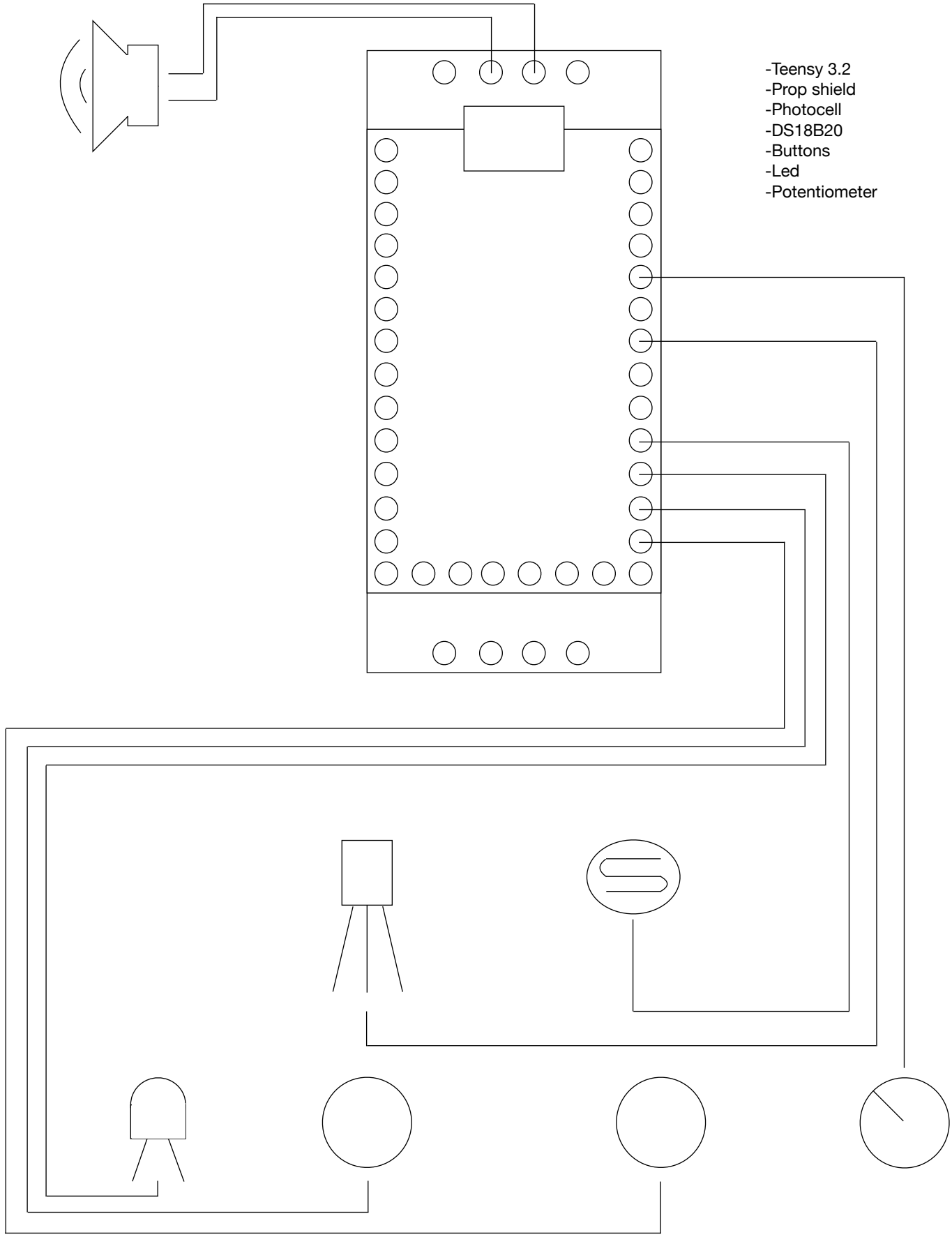
-Teensy 3.2
-Prop shield
-Photocell
-DS18B20
-Buttons
-Led
-Potentiometer

```
#include <EEPROM.h>
#include <OneWire.h>

OneWire ds(20);

int Button = 14;
int Bstate = 0;
int numberOfBPresses = 0;
int Bcount;
int triggerState = 0;
int Led = 15;
int Pbutton = 17;
int Pstate = 0;
int PtriggerState = 0;
int numberOfPPresses = 0;
int Pcount;
int readings[100];
int Pfreq = 0;
int Areading[100];

//int Temp = 20;
int tempValue = 0;
int tempReading = 0;
int tempFreq = 0;

//motionf
int rollFreq = 0;
int pitchFreq = 0;
int yawFreq = 0;



int amPot = 22;
float amReading = 0;
float level = 0;



unsigned int addr = 0;
int address = 0;
byte value;


#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

// GUItool: begin automatically generated code
AudioSynthWaveform waveform1;      //xy=102,348
AudioSynthWaveform waveform2;      //xy=105,405
AudioSynthWaveformSineModulated sine_fm1;       //xy=147,527
AudioSynthWaveform waveform4;      //xy=148,603
AudioMixer4 mixer1;         //xy=295,391
AudioSynthWaveform waveform3;      //xy=300,517
AudioFilterStateVariable filter1;        //xy=432,393
AudioOutputAnalog dac1;           //xy=695,267
AudioConnection patchCord1(waveform1, 0, mixer1, 0);
AudioConnection patchCord2(waveform2, 0, mixer1, 1);
AudioConnection patchCord3(sine_fm1, 0, mixer1, 2);
AudioConnection patchCord4(waveform4, sine_fm1);
AudioConnection patchCord5(mixer1, 0, filter1, 0);
AudioConnection patchCord6(waveform3, 0, filter1, 1);
AudioConnection patchCord7(filter1, 0, dac1, 0);
// GUItool: end automatically generated code


// GUItool: end automatically generated code


#include <NXPMotionSense.h>
#include <Wire.h>
#include <EEPROM.h>

NXPMotionSense imu;
NXPSensorFusion filter;


void setup(){



pinMode(Button, INPUT);
pinMode(Pbutton, INPUT);
 pinMode(Led, OUTPUT);

Serial.begin(9600);

//audio


AudioMemory(10);
waveform1.begin(WAVEFORM_TRIANGLE);
waveform2.begin(WAVEFORM_SQUARE);
waveform3.begin(WAVEFORM_SINE);
waveform4.begin(WAVEFORM_SINE);
pinMode(5, OUTPUT);
digitalWrite(5,HIGH);
delay(10);

//motion

Serial.begin(9600);
 imu.begin();

filter.begin(100);
}


void loop()
{

//amplitude

amReading = analogRead(amPot);
level = amReading/ 4096;

//waveforms

waveform1.frequency(Pfreq);
waveform1.amplitude(level);
waveform2.frequency(rollFreq);
waveform2.amplitude(level*0.05);
waveform3.frequency(tempFreq);
waveform3.amplitude(level);
waveform4.frequency(pitchFreq);
waveform4.amplitude(level);
sine_fm1.frequency(yawFreq);
sine_fm1.amplitude(level);


// record toggle
Bstate = digitalRead(Button);

if (Bstate != triggerState){
  if(Bstate == 1){

numberOfBPresses += 1;
   Bcount = numberOfBPresses % 2;

Serial.println(Bcount);
   triggerState = Bstate;

   if (Bcount == 0){

digitalWrite(Led, LOW);
    }
   if(Bcount == 1){

digitalWrite(Led, HIGH);
   }
  }else{
   triggerState = Bstate;
  }
}

// play toggle
Pstate = digitalRead(Pbutton);

if (Pstate != PtriggerState){
  if(Pstate == 1){

numberOfPPresses += 1;
   Pcount = numberOfPPresses % 2;
//
Serial.println(Pcount);
   PtriggerState = Pstate;

    if (Pcount == 0){

Serial.println("Stop");

Serial.println(readings[100]);
    }
   if(Pcount == 1){

Serial.println("Play");
   }
  }else{
   PtriggerState = Pstate;
  }
}



//write photocell
if (Bcount == 1){

int Pval = analogRead(2) / 4;

//
EEPROM.write(addr,Pval);
//addr = addr + 1;
//  if(addr == EEPROM.length())
//    addr = 0;
//
//    delay(200);
//}

//readings[100] = Pval;
//readings[100] = map(Pval, 1, 1023, 1, 500);
//addr = addr + 1;
```

```
//if(addr ==
readings[100])
//  addr = 0;
Pfreq =
map(Pval, 1,
1023, 1, 1000);


//  Serial.print
(Pval);
//
waveform1.freq
uency(Pfreq);
//
waveform1.amp
litude(level);

  delay(10);
}



//motion

if (Bcount == 1){

  float ax, ay, az;
  float gx, gy,
gz;
  float mx, my,
mz;
  float roll, pitch,
heading;

  if
(imu.available())
{
    // Read the
motion sensors

imu.readMotion
Sensor(ax, ay,
az, gx, gy, gz,
mx, my, mz);

    // Update the
SensorFusion
filter

filter.update(gx,
gy, gz, ax, ay,
az, mx, my, mz);
//
//    // print the
heading, pitch
and roll
    roll =
filter.getRoll();
    pitch =
filter.getPitch();
    heading =
filter.getYaw();
//
Serial.print("Ori
entation: ");
//
Serial.print(head
ing);
//    Serial.print("
");
//
Serial.print(pitch
);
//    Serial.print("
");
//
Serial.println(roll
);

rollFreq =
map(roll, -180,
180, 1, 500);
pitchFreq =
map(pitch,
-180, 180, 1,
500);
yawFreq =
map(heading,
-180, 180, 1,
500);

//
waveform2.freq
uency(Rollfreq);
//
waveform2.amp
litude(level);
delay(10);
  }
}



//temp

if (Bcount == 1){


 byte i;
  byte present =
0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius,
fahrenheit;

  if ( !
ds.search(addr))
{
//
Serial.println("N
o more
addresses.");
//
Serial.println();

ds.reset_search
();
    delay(250);
    return;
  }

//
Serial.print("RO
M =");
  for( i = 0; i < 8;
i++) {
//    Serial.write('
');
//
Serial.print(addr
[i], HEX);
  }

  if
(OneWire::crc8(
addr, 7) !=
addr[7]) {
//
Serial.println("C
RC is not
valid!");
      return;
  }
//
Serial.println();

  // the first
ROM byte
indicates which
chip
  switch
(addr[0]) {
    case 0x10:
//
Serial.println("
Chip =
DS18S20");  //
or old DS1820
      type_s = 1;
      break;
    case 0x28:
//
Serial.println("
Chip =
DS18B20");
      type_s = 0;
      break;
    case 0x22:
//
Serial.println("
Chip =
DS1822");
      type_s = 0;
      break;
    default:
//
Serial.println("D
evice is not a
DS18x20 family
device.");
      return;
  }

  ds.reset();

ds.select(addr);
  ds.write(0x44,
1);        // start
conversion, with
parasite power
on at the end

  delay(10);     //
maybe 750ms
is enough,
maybe not
  // we might do
a ds.depower()
here, but the
reset will take
care of it.

  present =
ds.reset();

ds.select(addr);
  ds.write(0xBE);
// Read
Scratchpad

//  Serial.print("
Data = ");
//
Serial.print(pres
ent, HEX);
//  Serial.print("
");
  for ( i = 0; i < 9;
i++) {           //
we need 9
bytes
    data[i] =
ds.read();
//
Serial.print(data[
i], HEX);
//    Serial.print("
");
  }
//  Serial.print("
CRC=");
//
Serial.print(One
Wire::crc8(data,
8), HEX);
//
Serial.println();


  // Convert the
data to actual
temperature
  // because the
result is a 16 bit
signed integer,
it should
  // be stored to
an "int16_t"
type, which is
always 16 bits
  // even when
compiled on a
32 bit
processor.
  int16_t raw =
(data[1] << 8) |
data[0];
  if (type_s) {
    raw = raw <<
3; // 9 bit
resolution
default
    if (data[7] ==
0x10) {
      // "count
remain" gives
full 12 bit
resolution
      raw = (raw &
0xFFF0) + 12 -
data[6];
    }
  } else {
    byte cfg =
(data[4] & 0x60);
    // at lower
res, the low bits
are undefined,
so let's zero
them
    if (cfg ==
0x00) raw = raw
& ~7;  // 9 bit
resolution,
93.75 ms
    else if (cfg ==
0x20) raw = raw
& ~3; // 10 bit
res, 187.5 ms
    else if (cfg ==
0x40) raw = raw
& ~1; // 11 bit
res, 375 ms
    //// default is
12 bit
resolution, 750
ms conversion
time
  }
  celsius =
(float)raw / 16.0;
  fahrenheit =
celsius * 1.8 +
32.0;
//  Serial.print("
Temperature =
");
//
Serial.print(celsi
us);
//  Serial.print("
Celsius, ");
//
Serial.print(fahre
nheit);
//
Serial.println("
Fahrenheit");

//fahrenheit =
tempReading;
tempFreq =
map(fahrenheit,
-67, 257, 1,
1023);
//tempValue=
map(tempReadi
ng, -67., 257.,
1, 1023.);
//tempFreq =
map(tempValue,
1, 1023, 1, 500);
//
waveform3.freq
uency(tempFreq
);
//
waveform3.amp
litude(level);
//Serial.print
(tempFreq);
//delay(10);
 }
}
```

Environmental Composition

Keanu yokoyama

Where you are right now,
Start recording.

For the next thirty seconds, think of a place to visit.
Stop recording.

Go to that place, and for the last thirty seconds of your journey,
Start recording.

Arrive at your destination.
Stop recording.

Find an area you find you find interesting, and for thirty seconds,
Start recording.

Observe your surrounds,
Stop recording.

Explore whats around you, and take a thirty second walk,
Start recording.

When you are finished,
Stop recording.