Analog input

- The Arduino's ATmega328 has 6 analog-to-digital (ADC) inputs, labeled A0 – A5. Each is essentially a voltmeter. The ADCs convert the voltages to numbers.
- The allowed voltage range is 0 V (ground) up to a specified voltage reference. The default voltage reference is the power supply, so the typical range is 0 V – 5 V.
- To get a voltage reading: `x = analogRead( pin )`, where pin is the specific analog input and x is and integer. There is no mode command for the analog inputs. The pin can designated simply as a number (0 – 5) or the "A" can be prepended (A0 – A5).
- The analog pins can also be used as digital inputs, for applications that needs lots of digital pins.
- As always, for further details, check the Arduino language reference.

Example

```
//Read the level of voltage of a sensor connected to
//analog pin 4. If it is higher than 3.5 V, turn on a
//warning LED connected to digital pin 6.

int sensorPin = A4;   //Note, the “A” is optional.
int ledPin = 6;
int x;
float voltage;

void setup()
{
  pinmode( ledPin, OUTPUT );
}

void repeat()
{
  x = analogRead( sensorPin );
  voltage = 5.0/1023.0*x;        //Convert number to volts
  if (voltage >= 3.5 )
    digitalWrite( ledPin, HIGH);
  else
    digitalWrite( ledPin, LOW );
  delay( 1000 );
}
```

Analog accuracy

- The ADCs in the Atmega328 are 10-bit, giving 2^{10} = 1024 voltage steps. An input of 0 V will correspond to "0" and "1023" will correspond to the maximum voltage (= reference voltage).
- An input voltage bigger than the reference will be "clipped" at 1023. Anything less than 0 V will be clipped at 0.
- With a 5-V reference, the resolution is (5 V)/1023 = 4.9 mV — not terribly precise but possibly good enough for many applications.
- The default reference is the power supply — either 5.0 V or 3.3 V.
- The measured voltage is only as accurate as the reference. Power supply voltages — either USB or using a voltage regulator — can easily vary by ±0.2 V (about 5%). Then measured analog voltage will be off by a similar amount.

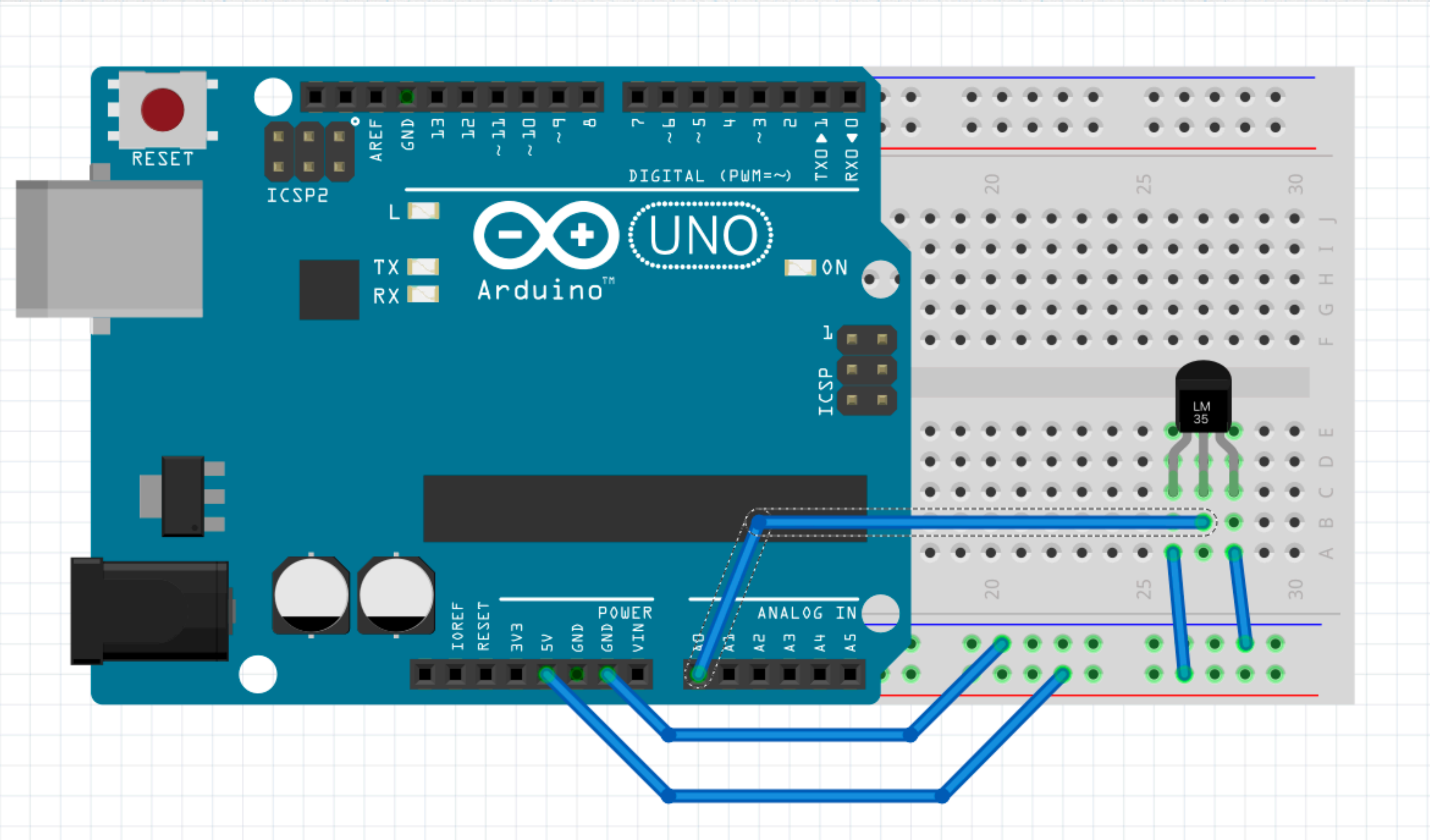
Accuracy and speed

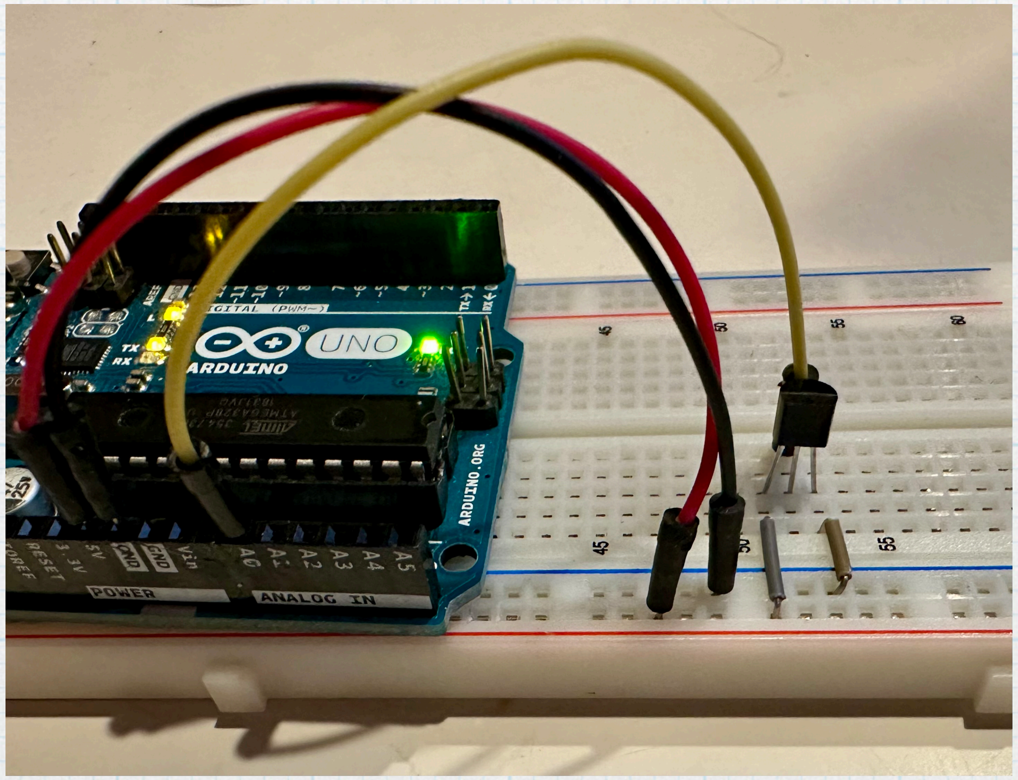
- There is a 1.10-V internal reference that can be used. This will give better resolution, (1.1 V)/1023 ≈ 1.1 mV, which might be advantageous when measuring small voltages from sensors, etc. However, the variance the internal reference it is about ±10%, so that is a disadvantage.
- It is possible to use attach a more accurate external reference. This would be attached to the AREF pin of the Atmega chip. An external reference could provide better accuracy and resolution.
- To use the internal reference, include the command `analogReference(INTERNAL)` in the setup function. For an external reference use `analogReference(EXTERNAL)` – and don't forget to attach the external reference voltage.
- The analog read time is about 100 microseconds, corresponding to a sample rate of 10 kHz — not terribly fast. Applying Nyquist, the highest frequency that could be reasonably sampled would be 5 kHz. This might be good enough for speech, but it is not inadequate for high fidelity audio.

Sensor example: LM35 temperature sensor

One of the most important uses for the analog inputs is in reading sensor voltages. There are *many* types of sensors that can be used with Arduino. (We will cover some in a later discussion.)

- One simple type of temperature sensor is the LM35 from Texas Instruments.
- 3 terminals: V_{CC}, ground, and v_{out}.
- 4 V < V_{CC} < 30 V, so works nicely with Arduino power supply.
- $v_{out} = (10\ \text{mV/°C})\cdot T$
- Room temp: ≈ 70°F = 21.1°C → v_{out} = 210 mV.
- Can measure below freezing if a negative supply is used as well.
- Data sheet: https://www.ti.com/lit/ds/symlink/lm35.pdf

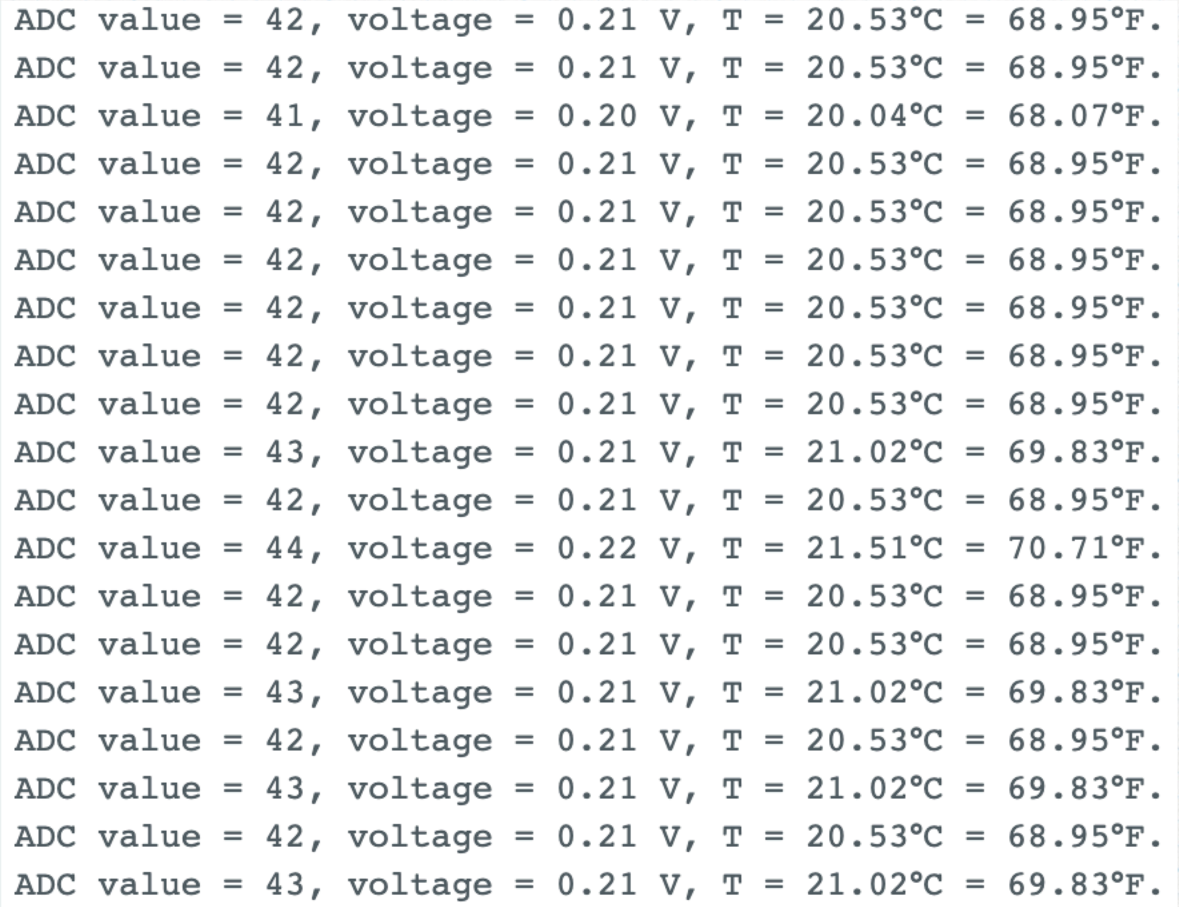




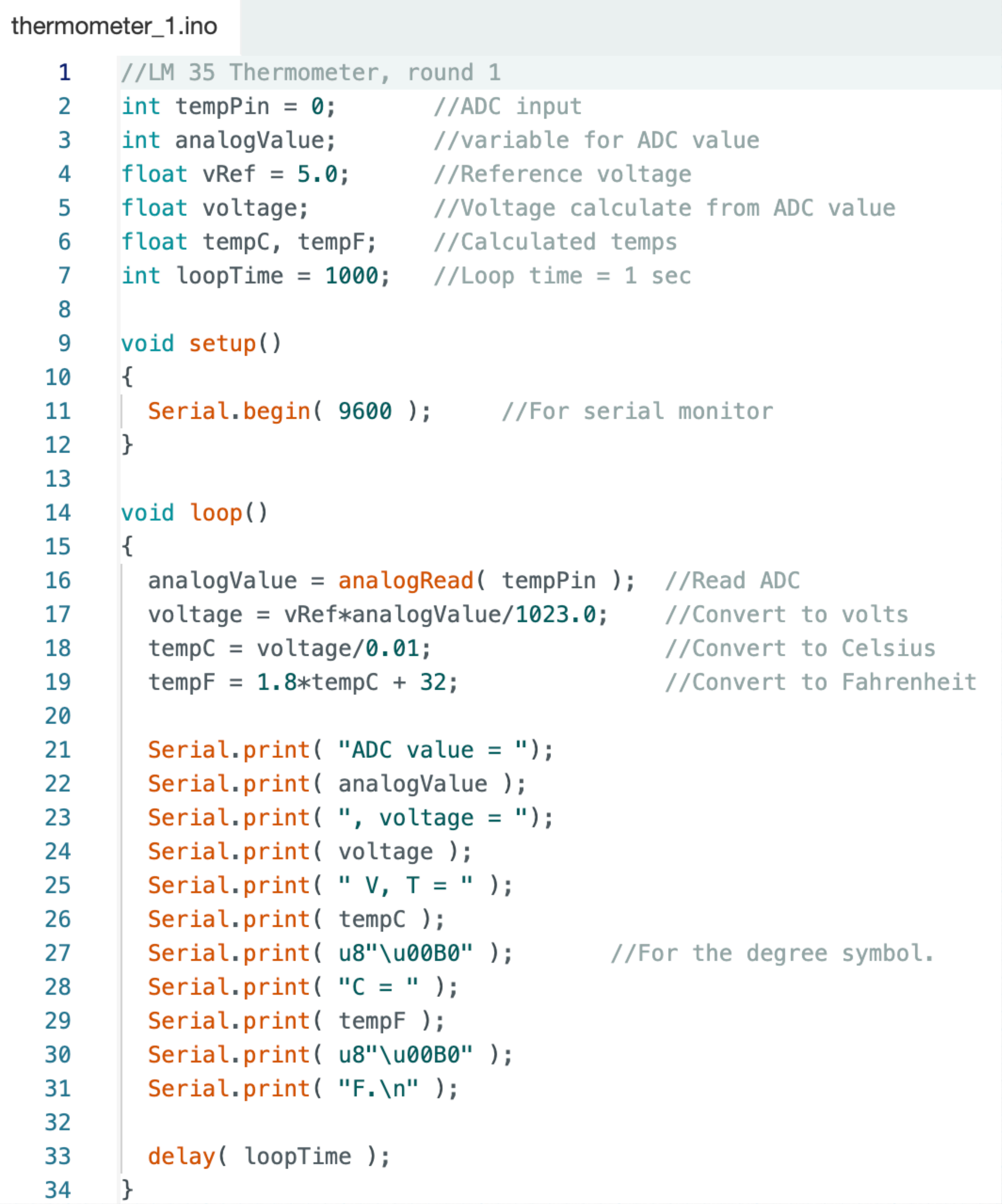
Use this setup to measure ambient air temperature.

LM35, first iteration

- Connect LM35 power and ground to Arduino. Use analog input 0 to read the sensor voltage.
- Use default $V_{CC} = 5$ V reference.
- Use serial monitor to display measurements.



```
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 41, voltage = 0.20 V, T = 20.04°C = 68.07°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 43, voltage = 0.21 V, T = 21.02°C = 69.83°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 44, voltage = 0.22 V, T = 21.51°C = 70.71°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 43, voltage = 0.21 V, T = 21.02°C = 69.83°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 43, voltage = 0.21 V, T = 21.02°C = 69.83°F.
ADC value = 42, voltage = 0.21 V, T = 20.53°C = 68.95°F.
ADC value = 43, voltage = 0.21 V, T = 21.02°C = 69.83°F.
```



```
//LM 35 Thermometer, round 1
int tempPin = 0;         //ADC input
int analogValue;         //variable for ADC value
float vRef = 5.0;        //Reference voltage
float voltage;           //Voltage calculate from ADC value
float tempC, tempF;      //Calculated temps
int loopTime = 1000;     //Loop time = 1 sec

void setup()
{
  Serial.begin( 9600 );     //For serial monitor
}

void loop()
{
  analogValue = analogRead( tempPin );  //Read ADC
  voltage = vRef*analogValue/1023.0;    //Convert to volts
  tempC = voltage/0.01;                 //Convert to Celsius
  tempF = 1.8*tempC + 32;               //Convert to Fahrenheit

  Serial.print( "ADC value = ");
  Serial.print( analogValue );
  Serial.print( ", voltage = ");
  Serial.print( voltage );
  Serial.print( " V, T = " );
  Serial.print( tempC );
  Serial.print( u8"\u00B0" );       //For the degree symbol.
  Serial.print( "C = " );
  Serial.print( tempF );
  Serial.print( u8"\u00B0" );
  Serial.print( "F.\n" );

  delay( loopTime );
}
```

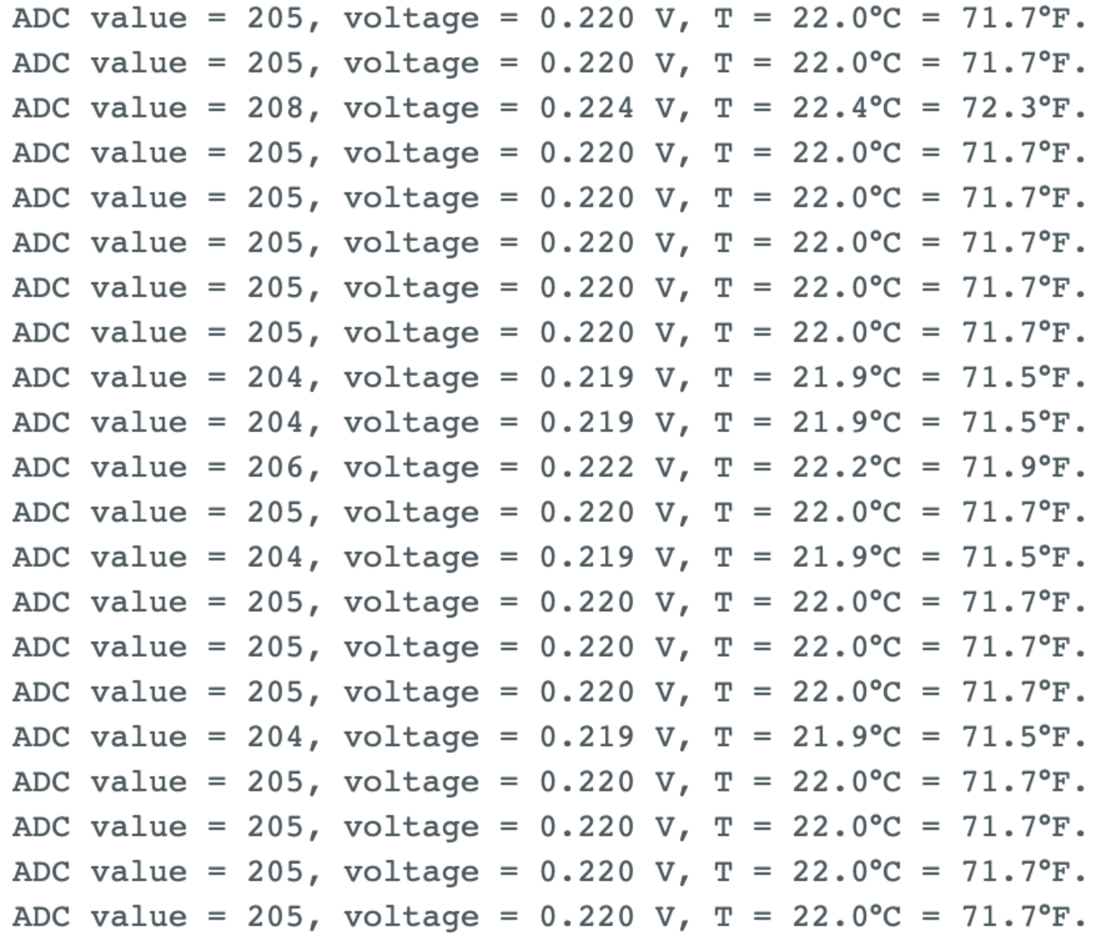
Seems easy enough.

LM35, first iteration

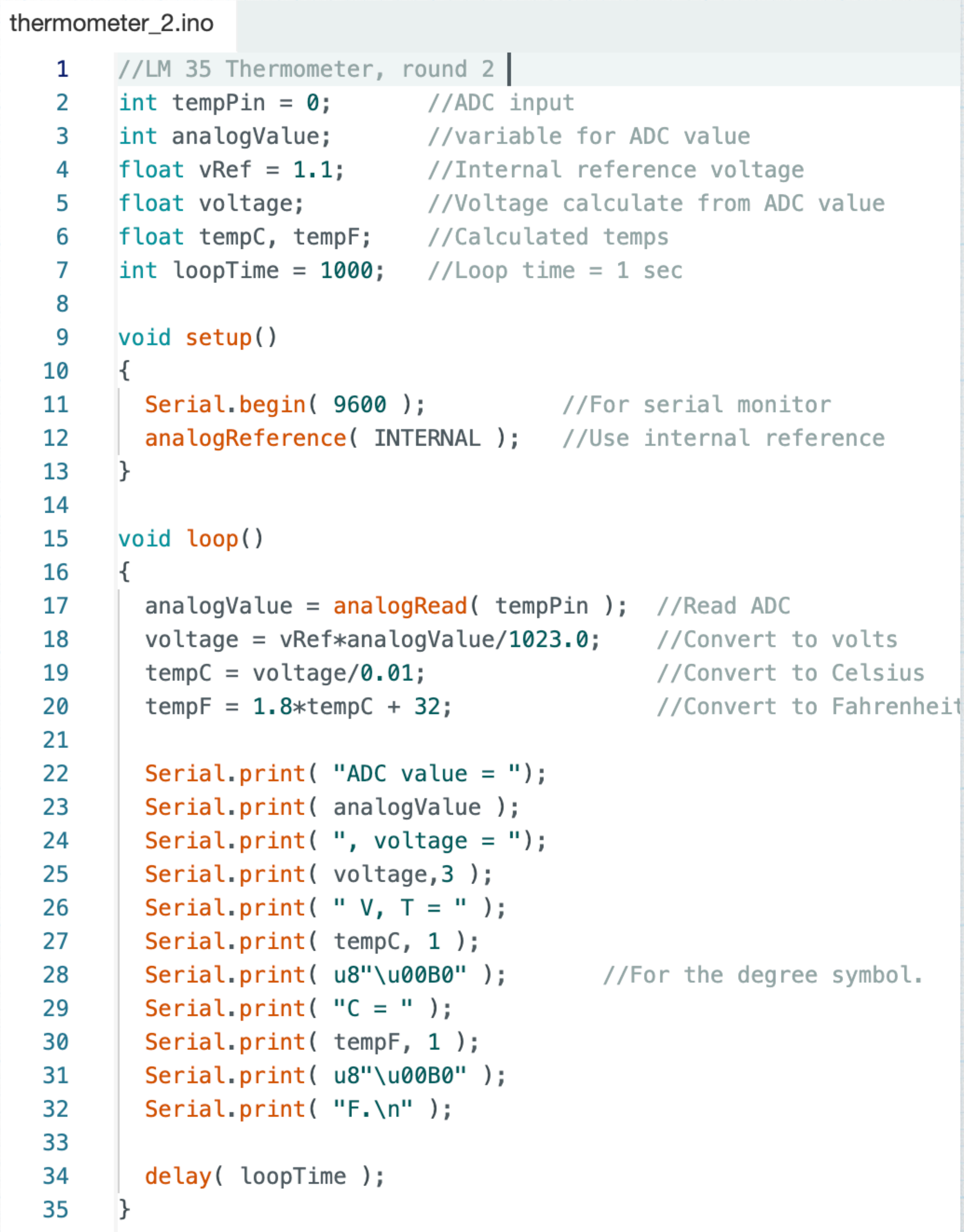
- In looking at the results of the first iteration, we note that the ADC is working at the very low end of the potential range. The values are around ≈ 42 out of 1023.
- With the 5-V reference, the voltage resolution is 4.9 mV, which corresponds to a temp resolution of about 0.5°C (0.9°F). Maybe we could better.
- From the snippet of results displayed (about 20 seconds worth), we see the range of values is ΔT = 1.5°C = 2.6°F. Seems like a lot for a quantity that really should not change much in such a short time.
- Temperature measured with a separate, calibrated thermometer was 71.0°F, so the LM 35 + Arduino temp seems a bit low.
- Reference voltage measured at the AREF pin was 5.08 V, a bit higher than expected.
- Might be able to improve resolution by using the internal reference. Add the line `analogReference(INTERNAL);` to the setup function.

LM35, second iteration – change reference voltage

- Switch to the internal reference Add command in setup function.
- Reference should be 1.1 V. Measured value was 1.077 V — close.
- Results are closer to the calibration, with less variance, $\Delta T = 0.6°\text{C} = 1.1°\text{F}$.



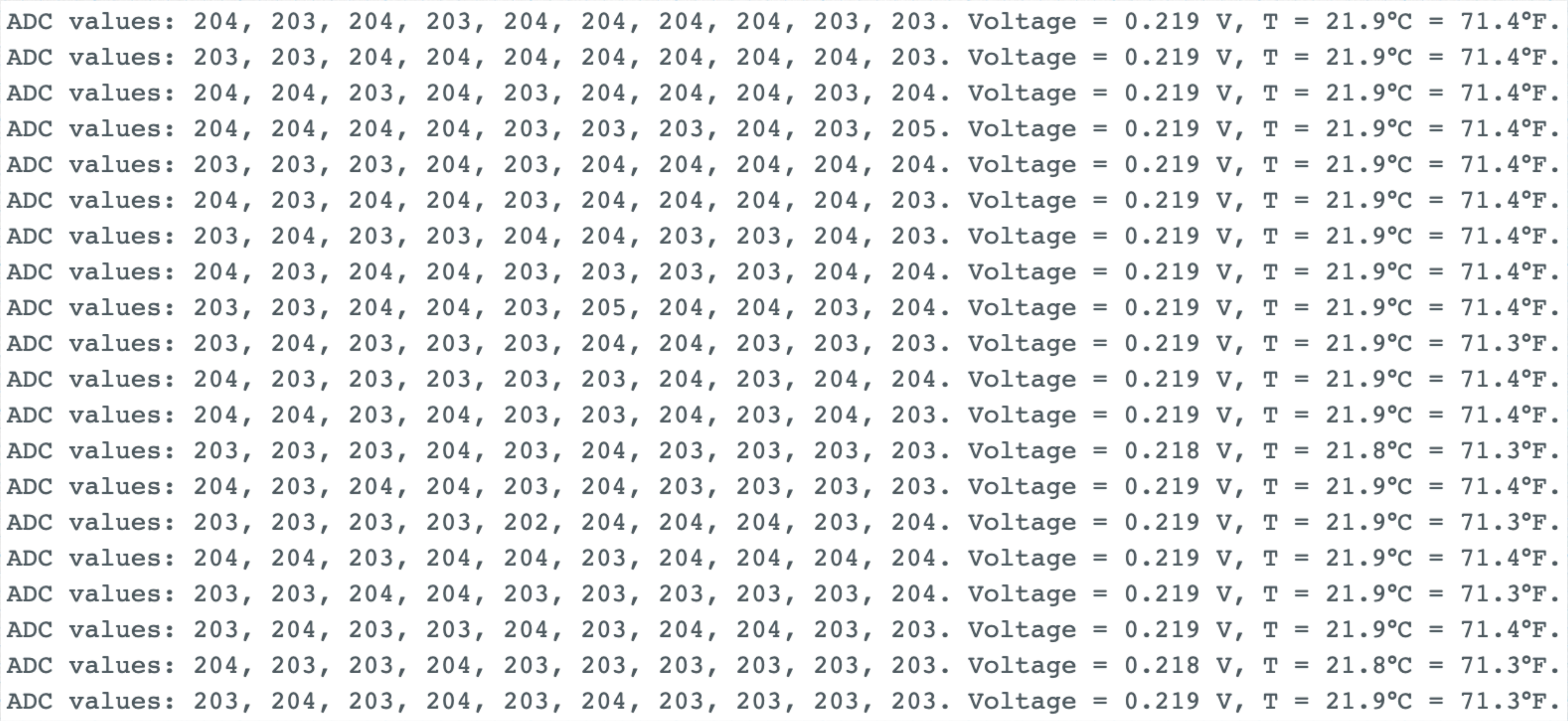
```
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 208, voltage = 0.224 V, T = 22.4°C = 72.3°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 204, voltage = 0.219 V, T = 21.9°C = 71.5°F.
ADC value = 204, voltage = 0.219 V, T = 21.9°C = 71.5°F.
ADC value = 206, voltage = 0.222 V, T = 22.2°C = 71.9°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 204, voltage = 0.219 V, T = 21.9°C = 71.5°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 204, voltage = 0.219 V, T = 21.9°C = 71.5°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
ADC value = 205, voltage = 0.220 V, T = 22.0°C = 71.7°F.
```



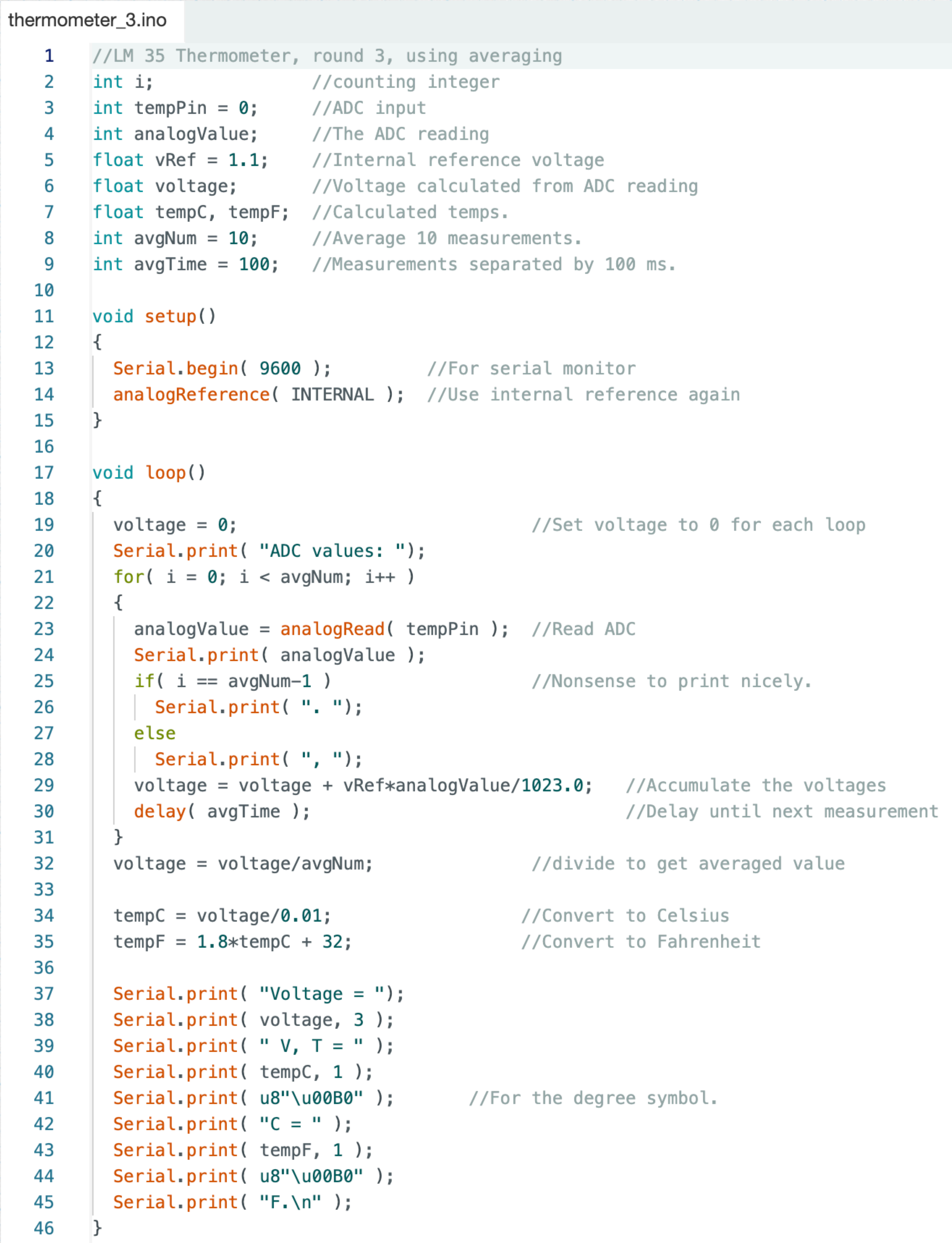
```
//LM 35 Thermometer, round 2
int tempPin = 0;       //ADC input
int analogValue;       //variable for ADC value
float vRef = 1.1;      //Internal reference voltage
float voltage;         //Voltage calculate from ADC value
float tempC, tempF;    //Calculated temps
int loopTime = 1000;   //Loop time = 1 sec

void setup()
{
  Serial.begin( 9600 );           //For serial monitor
  analogReference( INTERNAL );    //Use internal reference
}

void loop()
{
  analogValue = analogRead( tempPin );  //Read ADC
  voltage = vRef*analogValue/1023.0;    //Convert to volts
  tempC = voltage/0.01;                 //Convert to Celsius
  tempF = 1.8*tempC + 32;               //Convert to Fahrenheit

  Serial.print( "ADC value = ");
  Serial.print( analogValue );
  Serial.print( ", voltage = ");
  Serial.print( voltage,3 );
  Serial.print( " V, T = " );
  Serial.print( tempC, 1 );
  Serial.print( u8"\u00B0" );       //For the degree symbol.
  Serial.print( "C = " );
  Serial.print( tempF, 1 );
  Serial.print( u8"\u00B0" );
  Serial.print( "F.\n" );

  delay( loopTime );
}
```

LM35, third iteration – include averaging

- To improve accuracy further, use averaging. Make several measurements over a time interval and calculate the average value of those measurements.
- Works like a low-pass filter to reduce “high-frequency” variations.
- For example, measure 10 times in 1 sec (0.1 sec between measurements.
- Serial monitor results below. Code is on the next slide.
- Individual ADC values still vary somewhat. But, with averaging, the temps are much more steady — $\Delta T = 0.1°\mathrm{C} \approx 0.2°\mathrm{F}$.



```
ADC values: 204, 203, 204, 203, 204, 204, 204, 204, 203, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 204, 204, 204, 204, 204, 204, 204, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 204, 203, 204, 203, 204, 204, 204, 203, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 204, 204, 204, 203, 203, 203, 204, 203, 205. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 203, 204, 203, 204, 204, 204, 204, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 203, 204, 204, 203, 204, 204, 204, 204, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 204, 203, 203, 204, 204, 203, 203, 204, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 203, 204, 204, 203, 203, 203, 203, 204, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 204, 204, 203, 205, 204, 204, 203, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 204, 203, 203, 203, 204, 204, 203, 203, 203. Voltage = 0.219 V, T = 21.9°C = 71.3°F.
ADC values: 204, 203, 203, 203, 203, 203, 204, 203, 204, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 204, 203, 204, 203, 203, 204, 203, 204, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 203, 204, 203, 204, 203, 203, 203, 203. Voltage = 0.218 V, T = 21.8°C = 71.3°F.
ADC values: 204, 203, 204, 204, 203, 204, 203, 203, 203, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 203, 203, 202, 204, 204, 204, 203, 204. Voltage = 0.219 V, T = 21.9°C = 71.3°F.
ADC values: 204, 204, 203, 204, 204, 203, 204, 204, 204, 204. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 203, 203, 204, 204, 203, 203, 203, 203, 203, 204. Voltage = 0.219 V, T = 21.9°C = 71.3°F.
ADC values: 203, 204, 203, 203, 204, 203, 204, 204, 203, 203. Voltage = 0.219 V, T = 21.9°C = 71.4°F.
ADC values: 204, 203, 203, 204, 203, 203, 203, 203, 203, 203. Voltage = 0.218 V, T = 21.8°C = 71.3°F.
ADC values: 203, 204, 203, 204, 203, 204, 203, 203, 203, 203. Voltage = 0.219 V, T = 21.9°C = 71.3°F.
```



thermometer_3.ino

```
//LM 35 Thermometer, round 3, using averaging
int i;               //counting integer
int tempPin = 0;     //ADC input
int analogValue;     //The ADC reading
float vRef = 1.1;    //Internal reference voltage
float voltage;       //Voltage calculated from ADC reading
float tempC, tempF;  //Calculated temps.
int avgNum = 10;     //Average 10 measurements.
int avgTime = 100;   //Measurements separated by 100 ms.

void setup()
{
  Serial.begin( 9600 );         //For serial monitor
  analogReference( INTERNAL );  //Use internal reference again
}

void loop()
{
  voltage = 0;                            //Set voltage to 0 for each loop
  Serial.print( "ADC values: ");
  for( i = 0; i < avgNum; i++ )
  {
    analogValue = analogRead( tempPin );  //Read ADC
    Serial.print( analogValue );
    if( i == avgNum-1 )                   //Nonsense to print nicely.
      Serial.print( ". ");
    else
      Serial.print( ", ");
    voltage = voltage + vRef*analogValue/1023.0;   //Accumulate the voltages
    delay( avgTime );                              //Delay until next measurement
  }
  voltage = voltage/avgNum;               //divide to get averaged value

  tempC = voltage/0.01;                  //Convert to Celsius
  tempF = 1.8*tempC + 32;                //Convert to Fahrenheit

  Serial.print( "Voltage = ");
  Serial.print( voltage, 3 );
  Serial.print( " V, T = " );
  Serial.print( tempC, 1 );
  Serial.print( u8"\u00B0" );      //For the degree symbol.
  Serial.print( "C = " );
  Serial.print( tempF, 1 );
  Serial.print( u8"\u00B0" );
  Serial.print( "F.\n" );
}
```

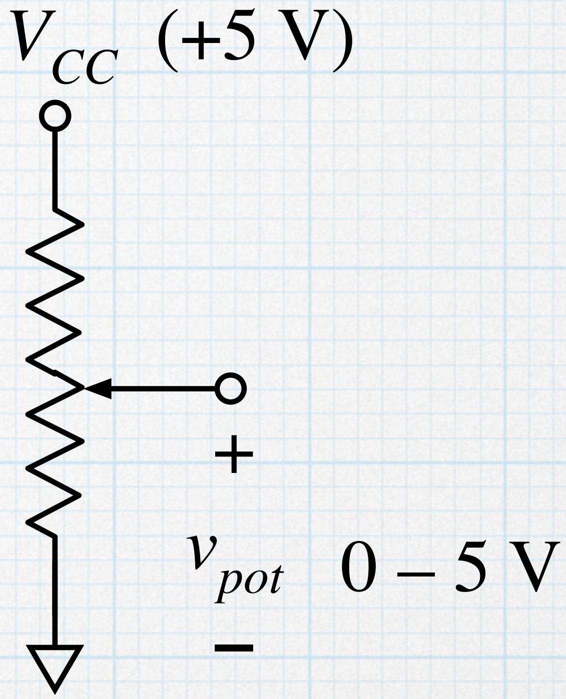
Moral of story:

- Using analog inputs is easy.
- Be mindful of voltage limits.
- Use a smaller reference voltage to improve resolution. But this also reduces the maximum range of voltages that can be measured.
- Use averaging to reduce random fluctuations. (Noise.)

A potentiometer as a variable input

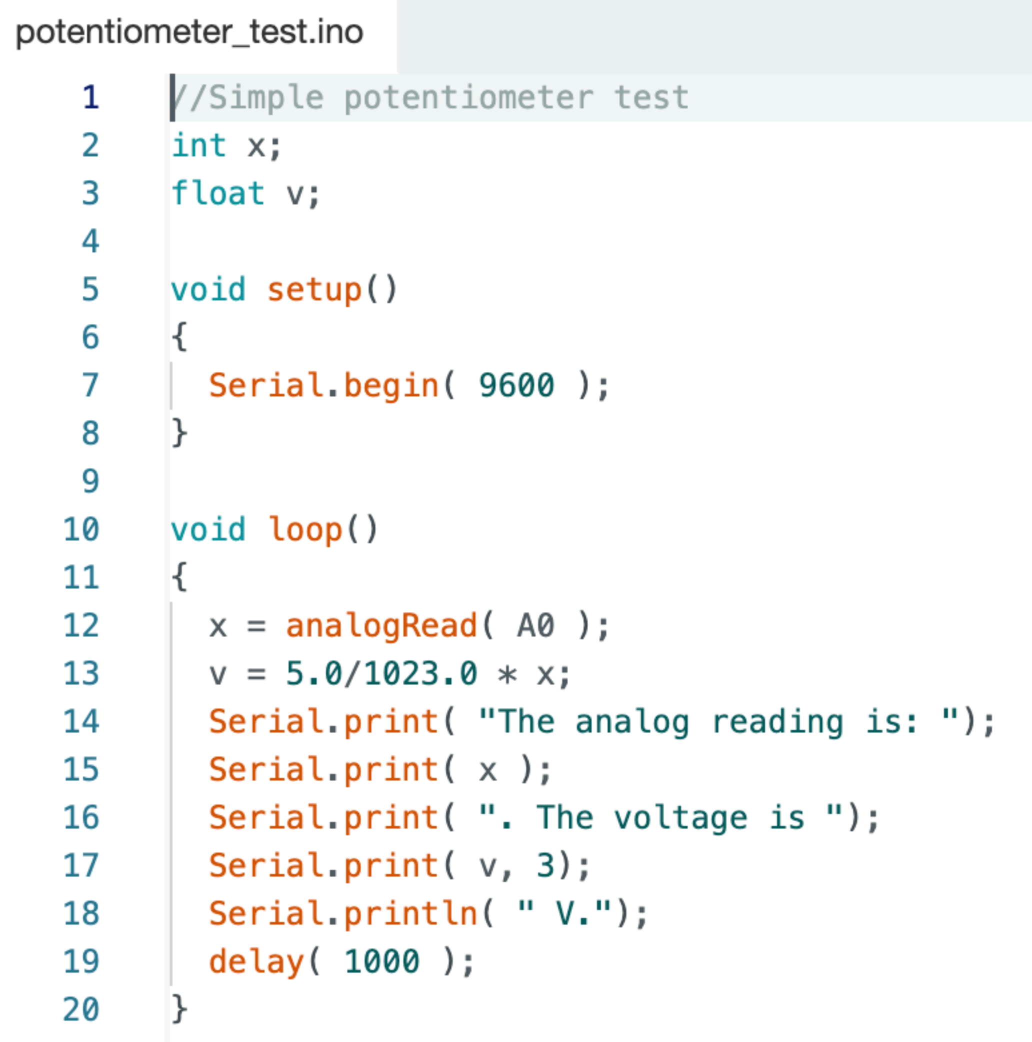
A potentiometer is a simple way to adjust parameters in a system. The potentiometer can swing from 0 to “full scale”, and the voltage can read at one of the analog inputs.

The value of the potentiometer is not important — 10 kΩ or 100 kΩ is fine.



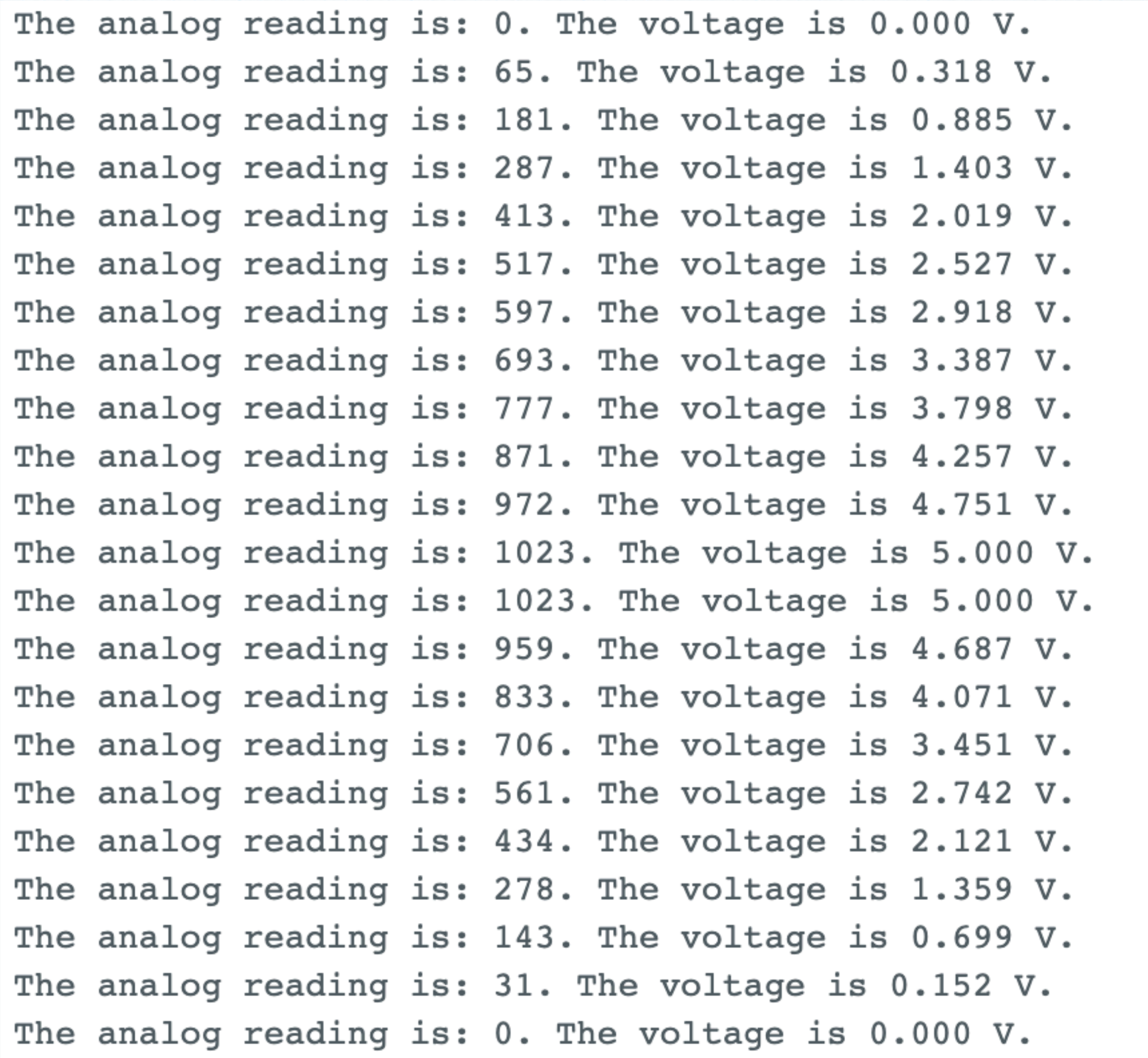
As a simple test, connect the outer leads of a potentiometer to the power supply and ground and the connect to the wiper to to analog input A0. Adjust the potentiometer and print the readings.

Potentiometer test



potentiometer_test.ino

```
//Simple potentiometer test
int x;
float v;

void setup()
{
  Serial.begin( 9600 );
}

void loop()
{
  x = analogRead( A0 );
  v = 5.0/1023.0 * x;
  Serial.print( "The analog reading is: ");
  Serial.print( x );
  Serial.print( ". The voltage is ");
  Serial.print( v, 3);
  Serial.println( " V.");
  delay( 1000 );
}
```

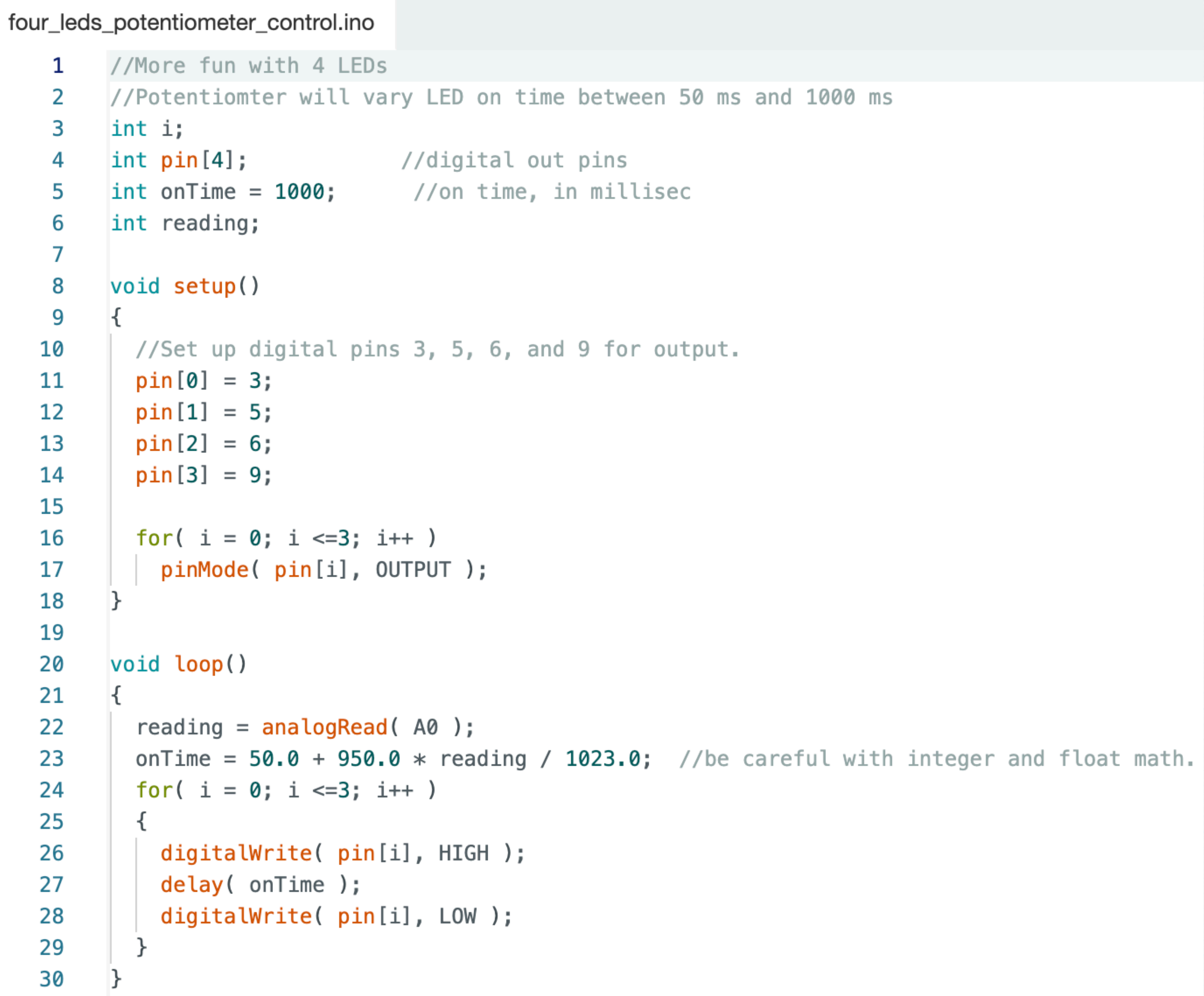


```
The analog reading is: 0. The voltage is 0.000 V.
The analog reading is: 65. The voltage is 0.318 V.
The analog reading is: 181. The voltage is 0.885 V.
The analog reading is: 287. The voltage is 1.403 V.
The analog reading is: 413. The voltage is 2.019 V.
The analog reading is: 517. The voltage is 2.527 V.
The analog reading is: 597. The voltage is 2.918 V.
The analog reading is: 693. The voltage is 3.387 V.
The analog reading is: 777. The voltage is 3.798 V.
The analog reading is: 871. The voltage is 4.257 V.
The analog reading is: 972. The voltage is 4.751 V.
The analog reading is: 1023. The voltage is 5.000 V.
The analog reading is: 1023. The voltage is 5.000 V.
The analog reading is: 959. The voltage is 4.687 V.
The analog reading is: 833. The voltage is 4.071 V.
The analog reading is: 706. The voltage is 3.451 V.
The analog reading is: 561. The voltage is 2.742 V.
The analog reading is: 434. The voltage is 2.121 V.
The analog reading is: 278. The voltage is 1.359 V.
The analog reading is: 143. The voltage is 0.699 V.
The analog reading is: 31. The voltage is 0.152 V.
The analog reading is: 0. The voltage is 0.000 V.
```

As a check, the potentiometer was adjusted until the reading was 511 = 2.502 V. Measuring the potentiometer voltage with a good voltmeter gave 2.539 V — close but not exact. The power supply voltage was measured to 5.08 V.

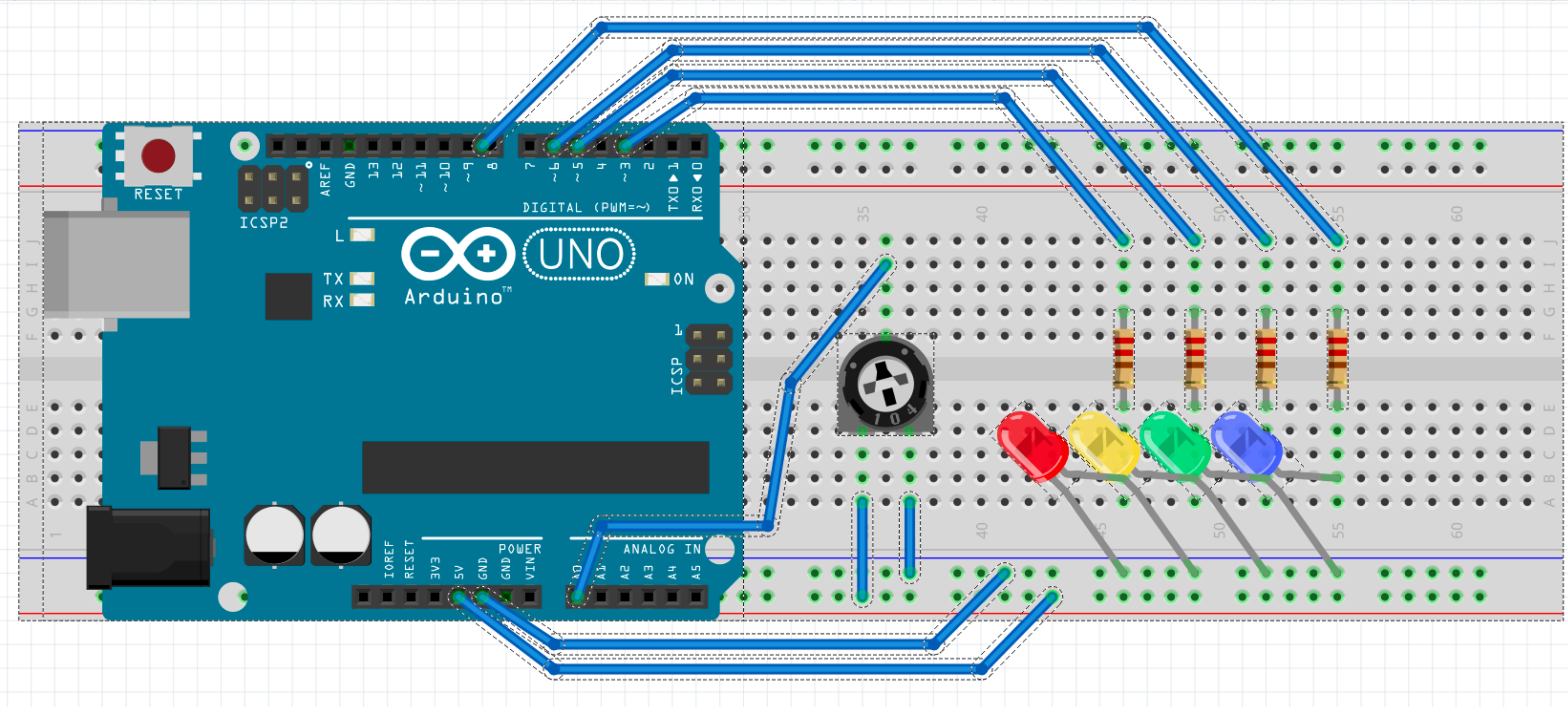
Example: Potentiometer to control the 4 LED pulse rate

Use the potentiometer to control the on time for the 4 LEDs (from digital output notes). Have it range between 50 ms and 1 s.



four_leds_potentiometer_control.ino

```
//More fun with 4 LEDs
//Potentiomter will vary LED on time between 50 ms and 1000 ms
int i;
int pin[4];            //digital out pins
int onTime = 1000;       //on time, in millisec
int reading;

void setup()
{
  //Set up digital pins 3, 5, 6, and 9 for output.
  pin[0] = 3;
  pin[1] = 5;
  pin[2] = 6;
  pin[3] = 9;

  for( i = 0; i <=3; i++ )
    pinMode( pin[i], OUTPUT );
}

void loop()
{
  reading = analogRead( A0 );
  onTime = 50.0 + 950.0 * reading / 1023.0;  //be careful with integer and float math.
  for( i = 0; i <=3; i++ )
  {
    digitalWrite( pin[i], HIGH );
    delay( onTime );
    digitalWrite( pin[i], LOW );
  }
}
```


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
104