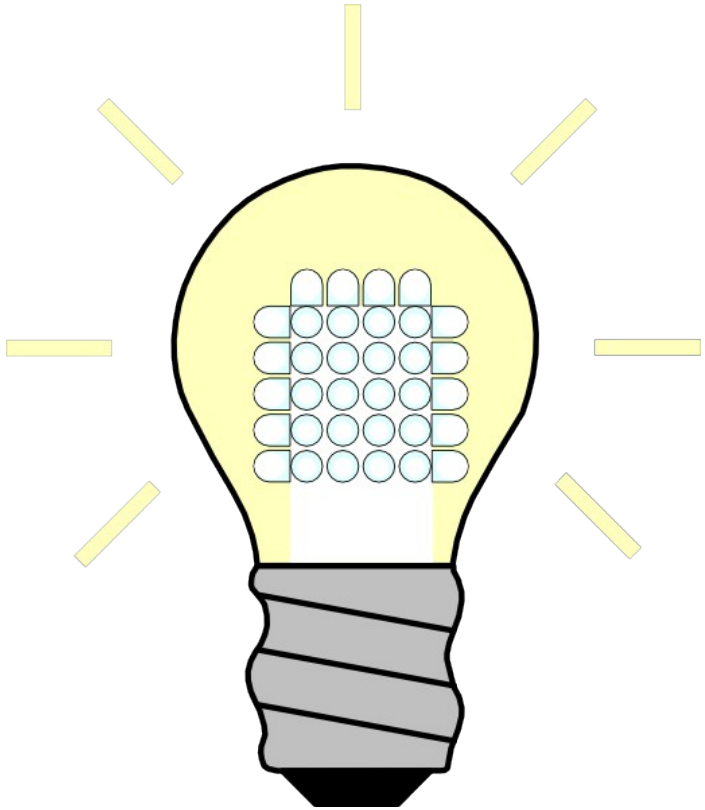
Making an LED Blink using Analogue Input

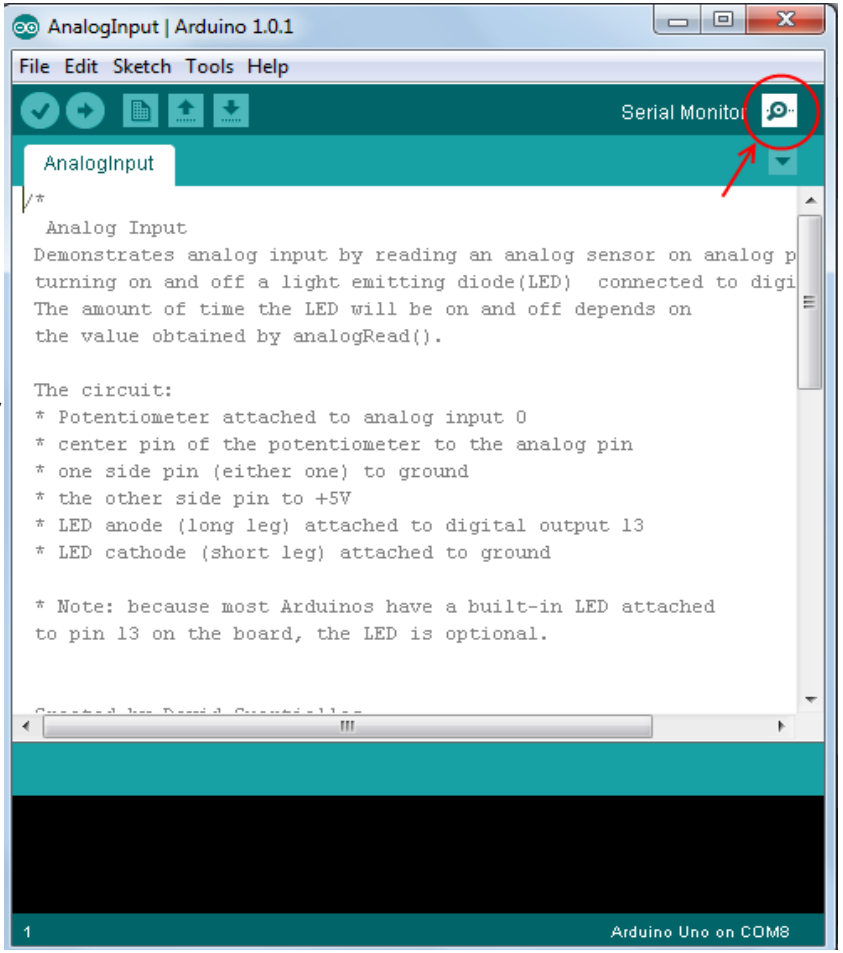


Created by Thomas Preece for the Technology Volunteers

Warwick Volunteers

Task

1. Create the circuit shown on next page
2. Open the Arduino IDE program on your computer
3. In the menu at the top of the Arduino program go to File → Examples → 03.Analog → AnalogInput
4. Change the code to look like the program code below, you should have to add two lines
5. Upload the code to the arduino
6. Turn the potentiometer and the light should blink either faster or slower
7. Open up the serial monitor (see picture to right)
8. Make sure that the bottom right box in the Serial Monitor is set to 9600 baud
9. You should see values between 1 and 1023 appearing on the screen, turn the potentiometer, the value should change. This value is the value we stored in the variable sensorValue and is the value that we are reading from the potentiometer.



Extensions

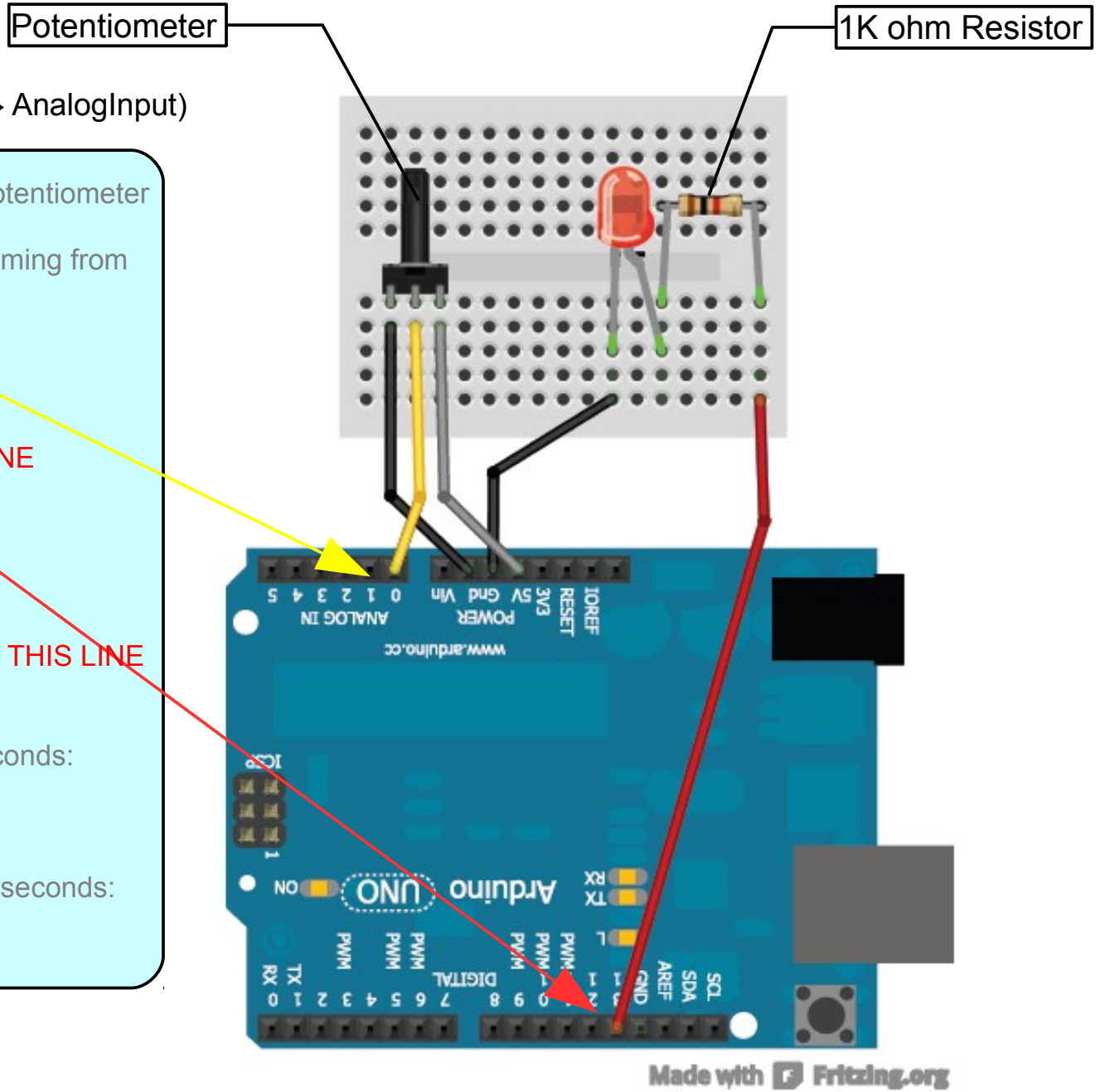
1. Change the code so when the value of potentiometer is above 500, the LED is on otherwise it is off.
   Hint: See the 'How to use the IF statement' section below
2. Change your circuit to use a light sensor (see circuit diagram on page 6)

Program Code: (File → Examples → 03.Analog → AnalogInput)

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from
the sensor

void setup() {
        // declare the ledPin as an OUTPUT:
        pinMode(ledPin, OUTPUT);
        Serial.begin(9600); //<------------ ADD THIS LINE
}

void loop() {
        // read the value from the sensor:
        sensorValue = analogRead(sensorPin);
        Serial.println(sensorValue);  //<------------ ADD THIS LINE
        // turn the ledPin on
        digitalWrite(ledPin, HIGH);
        // stop the program for <sensorValue> milliseconds:
        delay(sensorValue);
        // turn the ledPin off:
        digitalWrite(ledPin, LOW);
        // stop the program for for <sensorValue> milliseconds:
        delay(sensorValue);
}
```



Created by Thomas Preece for the Technology Volunteers

Warwick Volunteers

What is a variable?

A variable is like a number written on a blackboard, we can read what the number is and we can also change the number by erasing it and writing a new number onto the board. When ever we see the variable name in the code the value of that variable is used (reading the variable) unless we are using the special line:
variableName = [some value];
This line is used to change the number to a different number specified by what is written at [some value]'s position (writing the variable).
For example in our code above we use a variable called sensorValue. So in the code above at each loop the code reads the value of the potentiometer using analogRead(sensorPin) and writes that number to our variable named sensorValue. Lets assume that analogRead(sensorPin) was 205, and so now sensorValue has value 205. We then read the value of sensorValue and send it to the computer using **Serial**.println(sensorValue) so the computer will receive the value '205'. We finally delay the program by sensorValue number of milliseconds (1000 milliseconds = 1 second) using delay(sensorValue); so the program will wait 205 milliseconds.

What do Serial.begin and Serial.println actually do?

Basically **Serial**.begin(9600) tells the Arduino to open communication with the Computer its connected to and at what speed we want the connection at, in this case our speed is 9600. Without this line in the code we wouldn't be able to use **Serial**.println() as the Arduino wouldn't know where to send the information.

Serial.println() allows us to send information to our computer and we can then view it on the Serial Monitor. In our code we send the value contained in the variable sensorValue using the line **Serial**.println(sensorValue) but we can also send other things such as Serial.println("hello") which sends the text 'hello' to the computer.

How to use the IF statement

Its General form is:

```
if(Statement){
        Code here is run when statement is true/correct
}else{
        Code here is run when statement is false/wrong
}
```

The else bit is optional so we could just have:

```
if(Statement){
        Code here is run when statement is true/correct
}
```

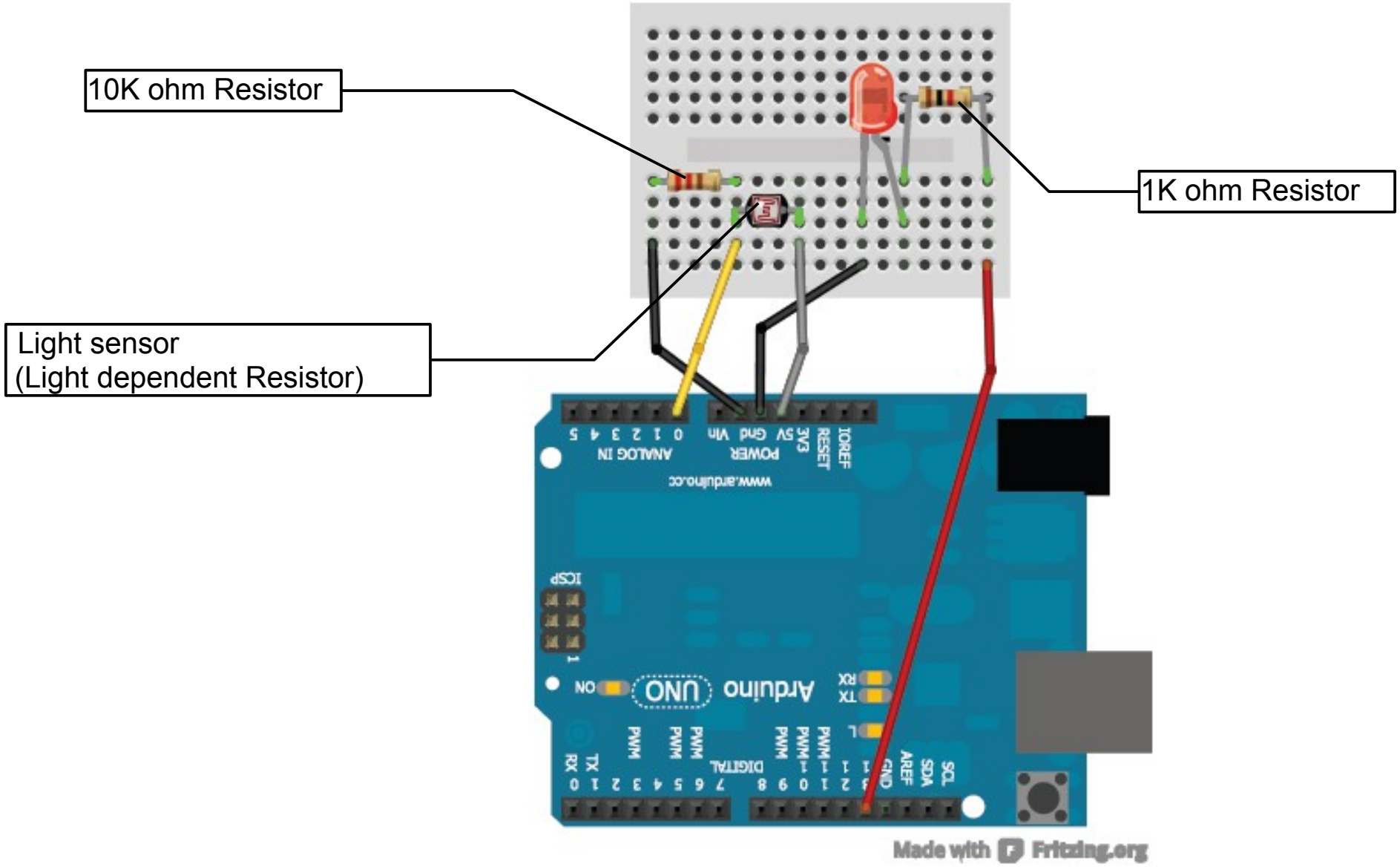
Statement is something that is either true or false for example:

5 > 3 is true because 5 is larger than 3.
5 < 3 is false because 5 is NOT smaller than 3.
However we are not just limited to numbers in our statement, we may use variables too. For example sensorValue < 200 is true if variable 'sensorValue' has value 105 or any value smaller than 200. This would be the code:

```
if(sensorValue < 200){
        Code here is run when 'sensorValue' has value smaller than 200
}
```

We may also use the double equals operator == in a statement. This operator checks if the two are the same value and returns true otherwise it returns false. For example:
5 == 3 is false because 5 is clearly not the same as 3.
5 == 5 is true because 5 is the same as 5
sensorValue == 5 is true only if sensorValue has value 5.

Analogue Input using a Light Dependent Resistor



Created by Thomas Preece for the Technology Volunteers

Warwick Volunteers