Rodni

What will yours be?

VERSION 4

Welcome to Rodni, a modular animatronic animal of your own creation for learning how easy it is to enter the world of software programming and micro controllers.

During the Rodni workshop you will build your own Rodni and learn how to make his heart "beat", his head move and also give him sight and the ability to interact with you.

The Rodni project is not about learning a programming language but about learning how easy it is to get quick results using material downloaded from the internet and picking up the necessary skills as you progress.

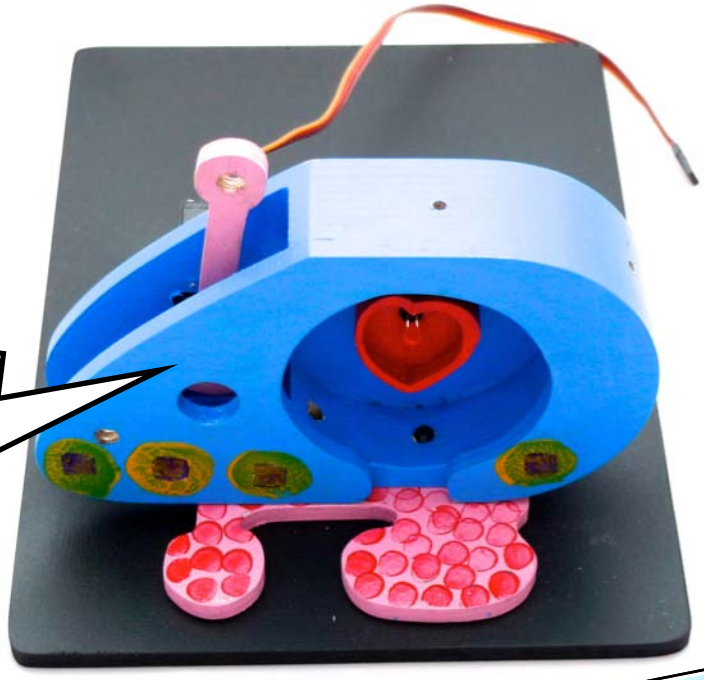
Rodni is powered by Arduino. This is an inexpensive micro controller, a small computer. It uses free software downloaded onto your laptop or PC. Once the Arduino has been programmed it will run independently of your main computer enabling you to create many exciting projects. Look at www.instructables.com for examples of what is possible.

More details about Rodni and many resources can be found at www.lowbot.co.uk

Rodni was created by Philip Lowndes

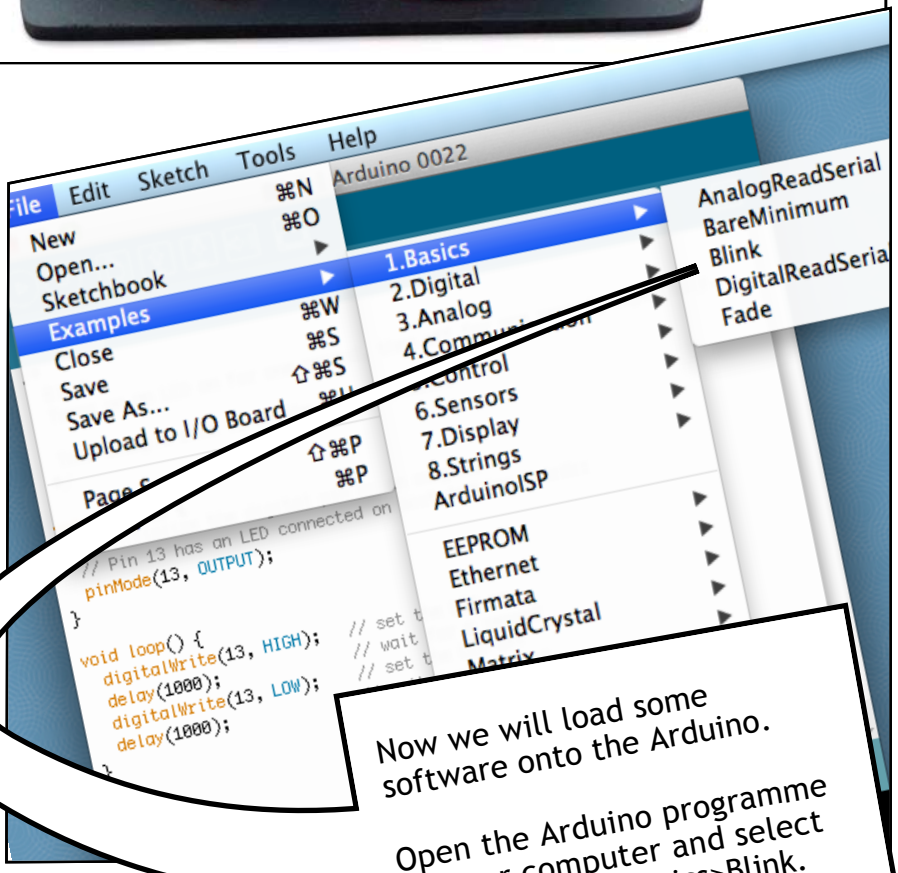
Making Rodni's heart beat

1



Choose a baseboard, a set of feet, a body and a heart. Assemble as shown.

2



Now we will load some software onto the Arduino.

Open the Arduino programme on your computer and select File>Examples>Basics>Blink.

3

You can now see the Blink programme. Not important at this stage but it has 3 parts:

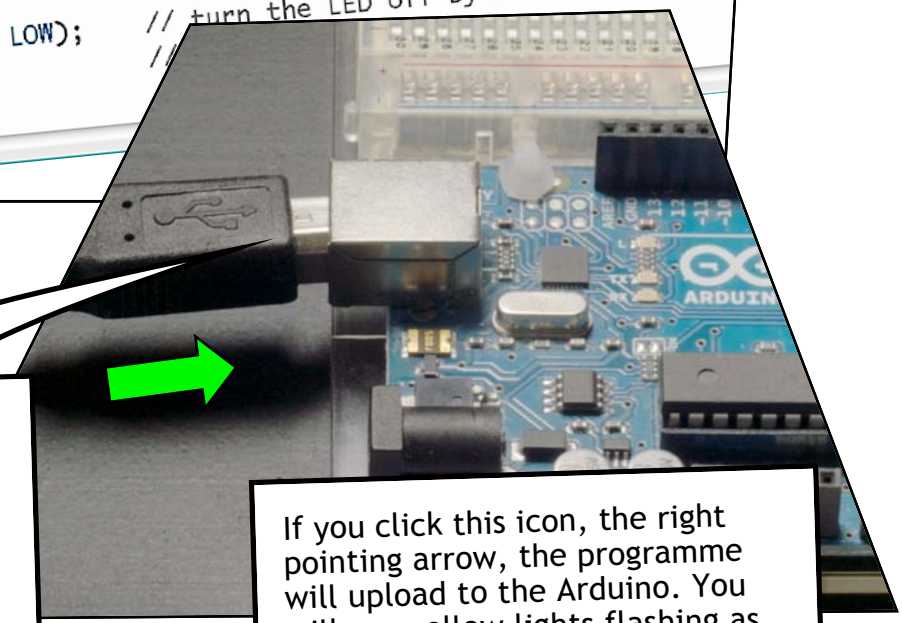
1) A list of variables. Here int led=13;
2) void setup() that includes things that the programme should read only once.
3) void loop() that will keep repeating while the programme runs.

Notice that some lines of the programme have // in front of them. This tells the Arduino not to read these lines and they are used for making notes in the programme to make it easier to understand what each part does.

```
Blink
Turns on an LED on for one secon
This example code is in the publ
*/
// Pin 13 has an LED connected on
// give it a name:
int led = 13;
// the setup routine runs once w
void setup() {
  // initialize the digital pin
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the volta
  delay(1000);
}
```

4

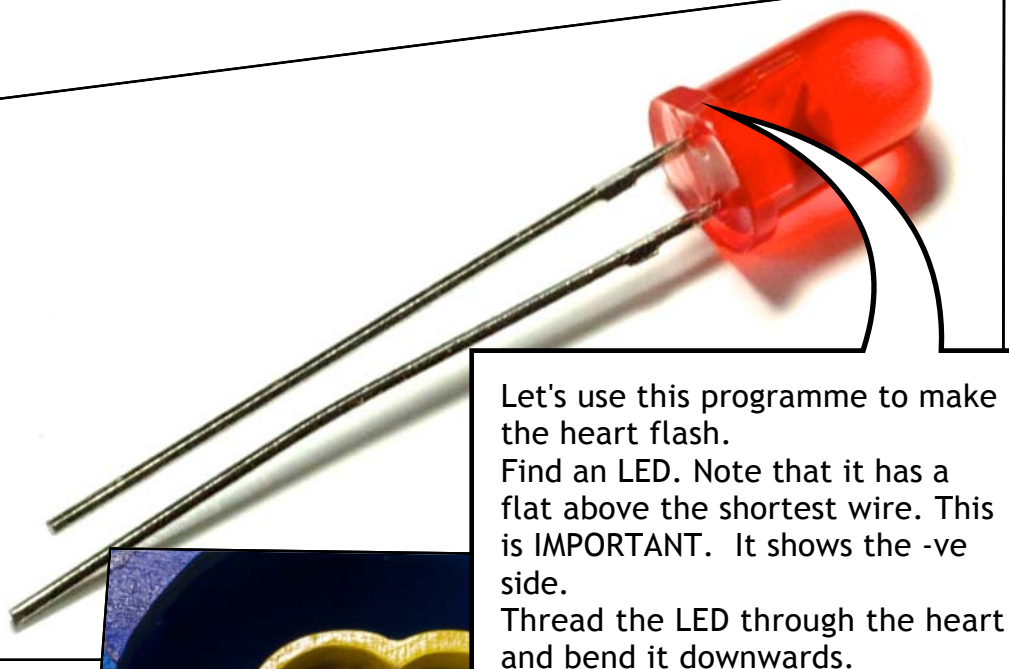
Plug the USB lead into the Arduino board and into your computer.
Go to Tools>Board and select your board. It is probably a UNO.
Now go to Tools>Serial Port and select a USB port.



If you click this icon, the right pointing arrow, the programme will upload to the Arduino. You will see yellow lights flashing as the computer talks to the Arduino.
Eventually the programme will run and an LED will steadily flash on the Arduino.

5

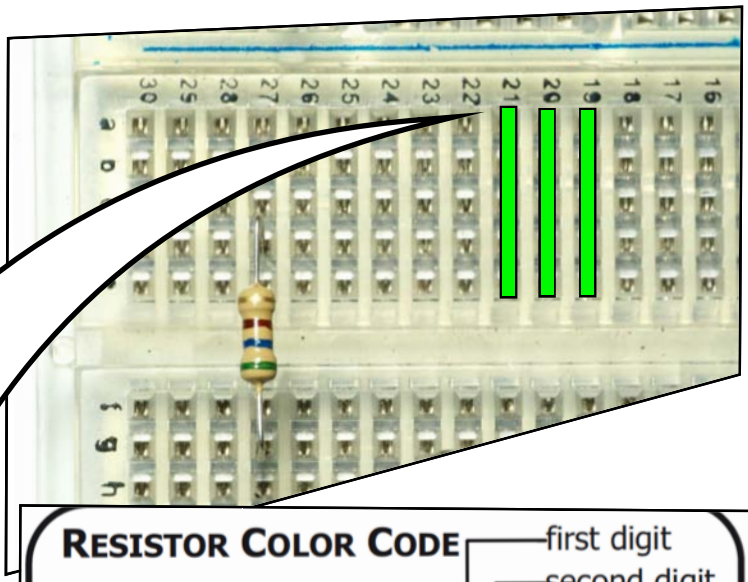
6

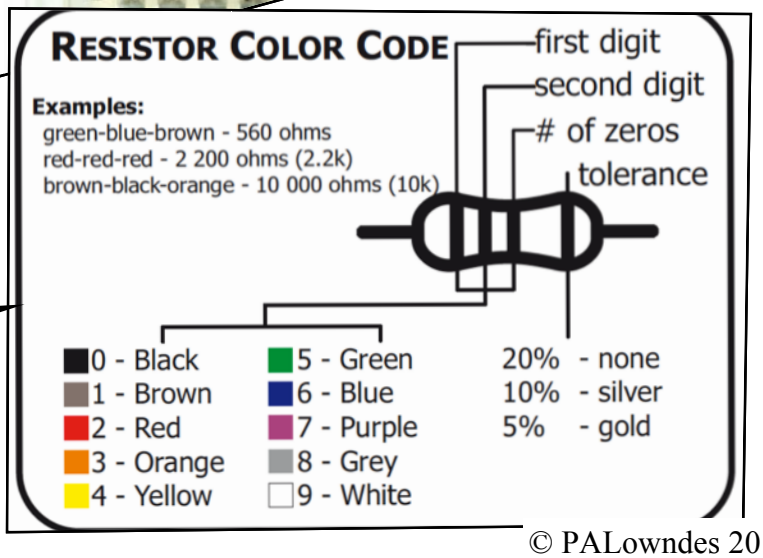


Let's use this programme to make the heart flash.
Find an LED. Note that it has a flat above the shortest wire. This is IMPORTANT. It shows the -ve side.
Thread the LED through the heart and bend it downwards.

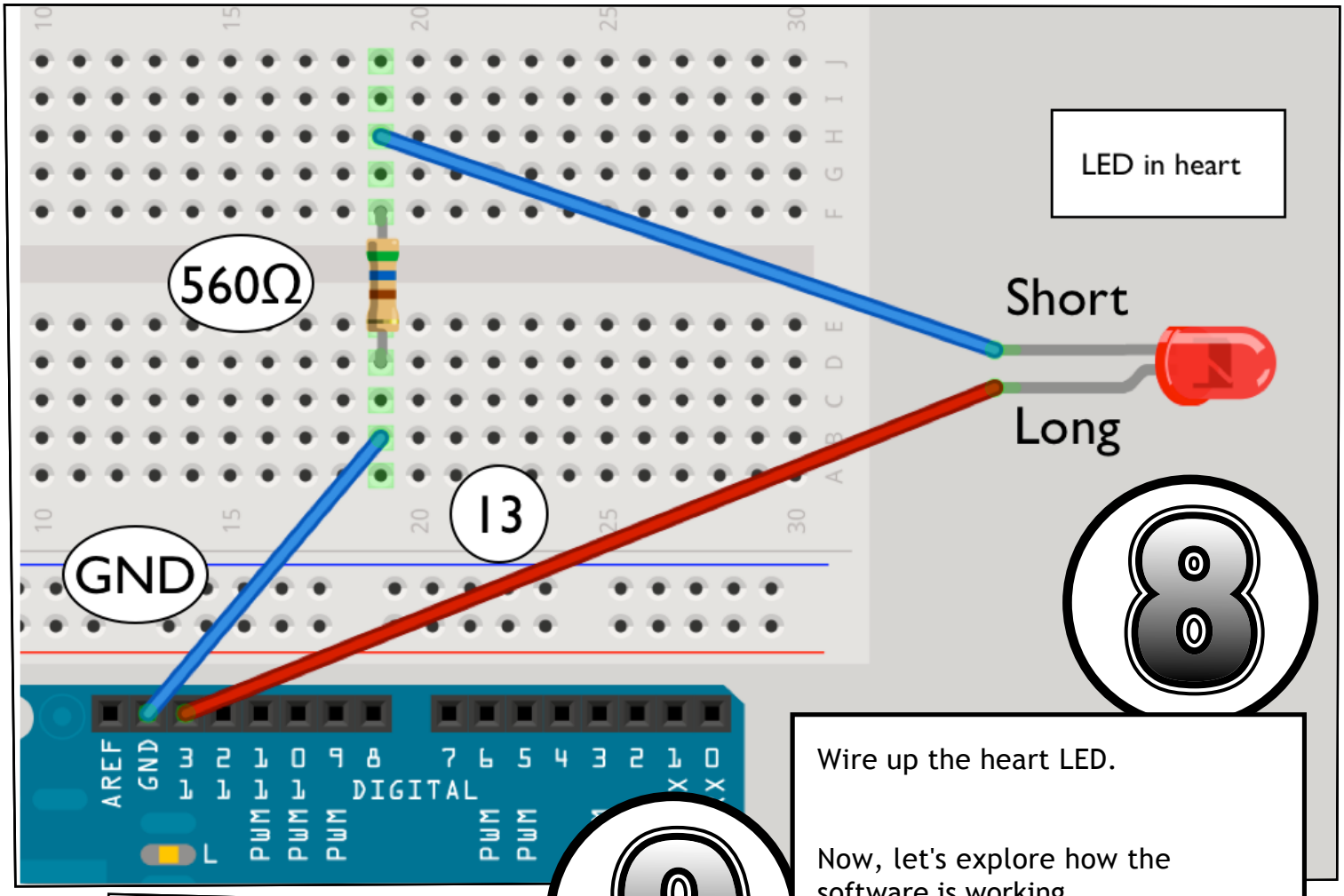
7

Now find a 560 ohm resistor. These are colour coded green, blue, brown, gold. Place it in the breadboard. 2 possible positions are shown in the photograph. The pins in each line are connected as shown so the electronics must bridge them.
We are using the resistor to reduce the amount of current reaching the LED to prevent it from burning out.





WHAT VALUE WOULD BROWN, BLACK, ORANGE, GOLD BE?



Wire up the heart LED.

Now, let's explore how the software is working.
The words in a line following // are ignored by the Arduino. These are just notes explaining the program

```
*/

// Pin 13 has an LED connected on most
// give it a name:
int led = 13;

// the setup routine runs once when you press reset
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is th
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by maki
  delay(1000);               // wait for a second
}
```

At the top int led = 13 is telling the Arduino that we are using pin 13 and calling it led.
The void setup() section tells the Arduino things that it only needs to know once. Here it is used to say that pin we called led is being used and pinmode OUTPUT tells it that it will send outgoing signals.
Notice the use of { }. These say that this is a discrete section of the code. The code will not run without them.

Void loop() is the section that the Arduino loops repeatedly.
Again note the use of { }
digitalWrite refers to whatever is in the brackets following ( ). Here it is saying to switch on the LED. HIGH means on.
delay tells it how long to stay on. In this case for 1000 milliseconds (1 second).
The next digitalWrite is to switch off again. LOW

Note the use of ; after each command. The programme will not run without this.
The words in brown are programming language words. If you enter them correctly the computer will colour them brown.

10

TRY CHANGING THE LED PIN NUMBER FROM 13 TO 8 IN THE PROGRAMME. NOW UPLOAD - WHAT HAPPENS?

THE LED NO LONGER FLASHES. YOU NEED TO MOVE THE WIRE FROM 13 TO 8 ON THE ARDUINO. NOW IT WILL FLASH.
YOU HAVE DONE YOUR FIRST BIT OF PROGRAMMING - EASY!

11

NOW FOR ANOTHER CHALLENGE.
CAN YOU MAKE THE LED FLASH MORE SLOWLY?

TRY ALTERING THE NUMBERS FOR DELAY AND THEN UPLOAD TO THE ARDUINO BY CLICKING THE UPLOAD ARROW.

12

BRILLIANT

NOW, YOU ARE ABLE MAKE THE LED BLINK AT A REGULAR FREQUENCY.

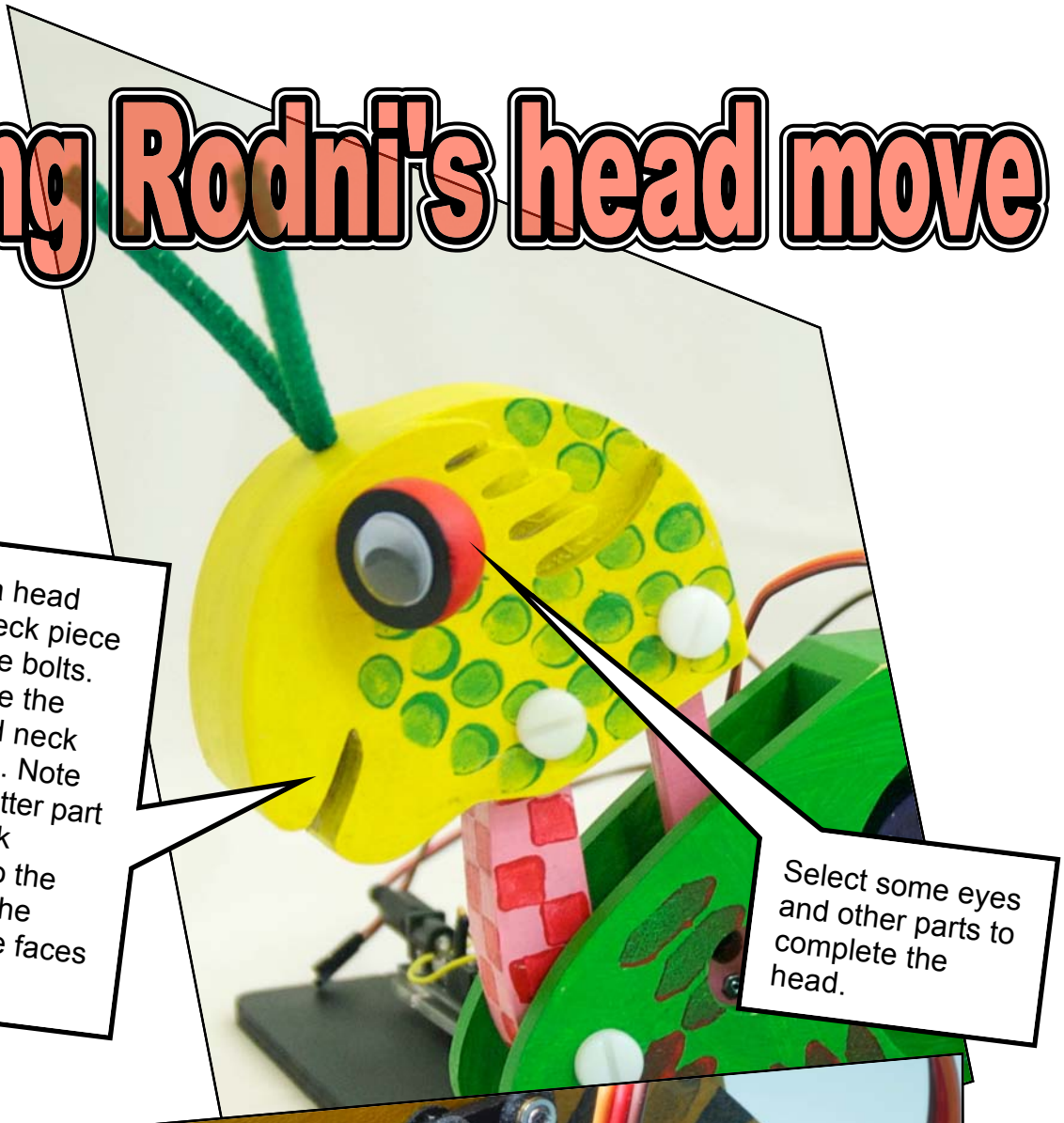
THE CHALLENGE NOW IS TO MAKE THE HEART BEAT.

HOW WILL YOU DO IT?

Making Rodni's head move

13

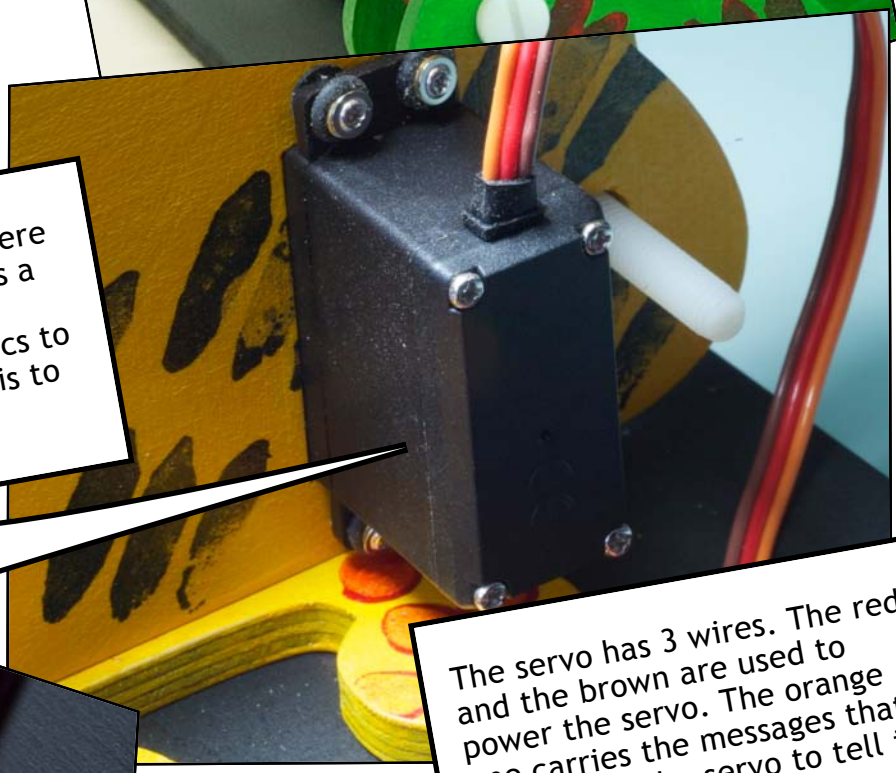
Select a head and a neck piece and three bolts. Assemble the head and neck as shown. Note that the fatter part of the neck attaches to the body and the curved side faces forwards.

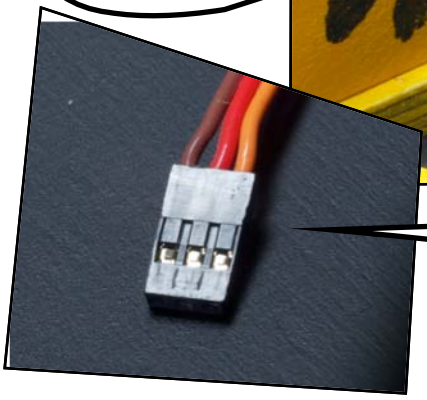


Select some eyes and other parts to complete the head.

14

At the back of the body there is a small black box. This is a servo. It contains a small motor and some electronics to control it. We will use this to make Rodni move.

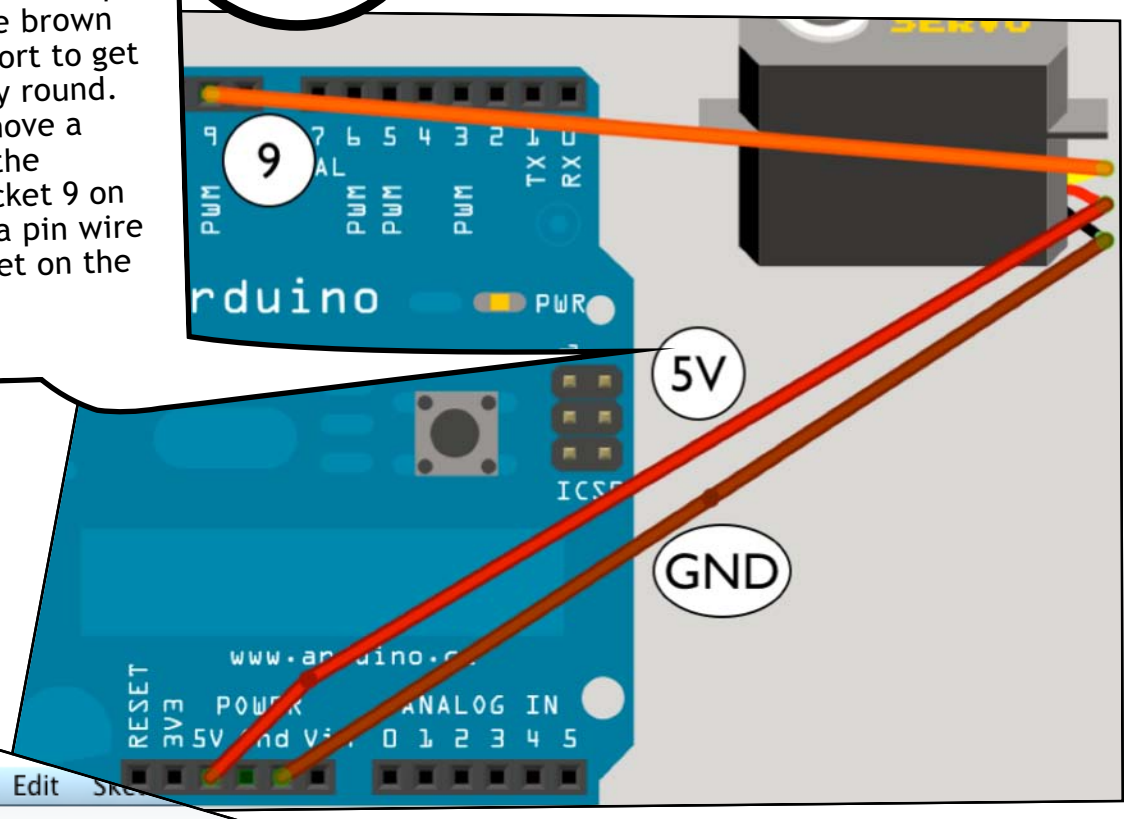


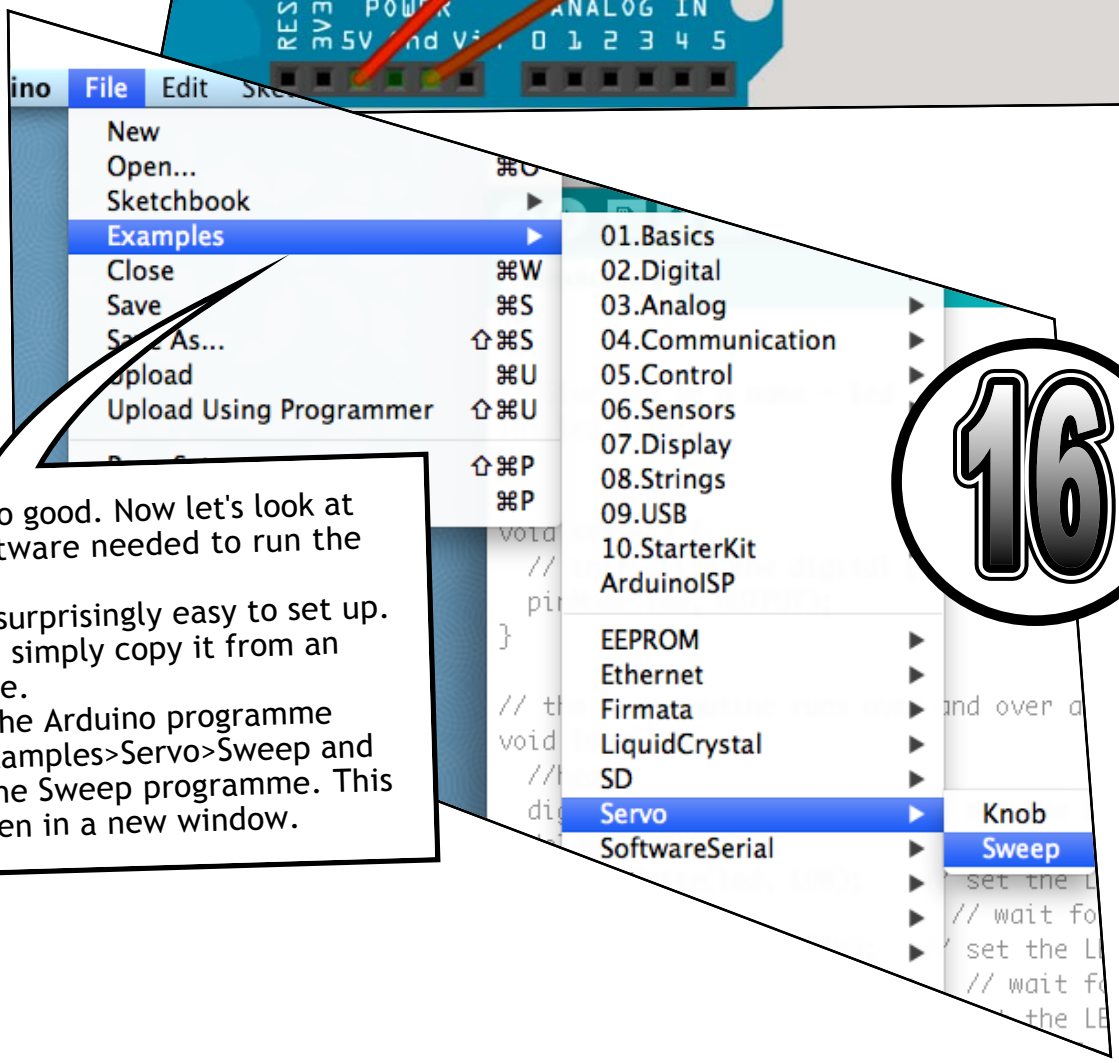


The servo has 3 wires. The red and the brown are used to power the servo. The orange one carries the messages that we send to the servo to tell it when to move.
On some servos these wires are red, black and white.

The servo is very simple to connect.
First let's give it some power. Run a pin wire from the red +ve strip on the breadboard to the servo plug socket that the red wire comes into. Then run a wire from the blue -ve strip to the plug where the brown wire runs in. It is import to get these the correct way round.
The servo may now move a little. Now let's add the control wire. Find socket 9 on the Arduino and run a pin wire from this to the socket on the orange wire.

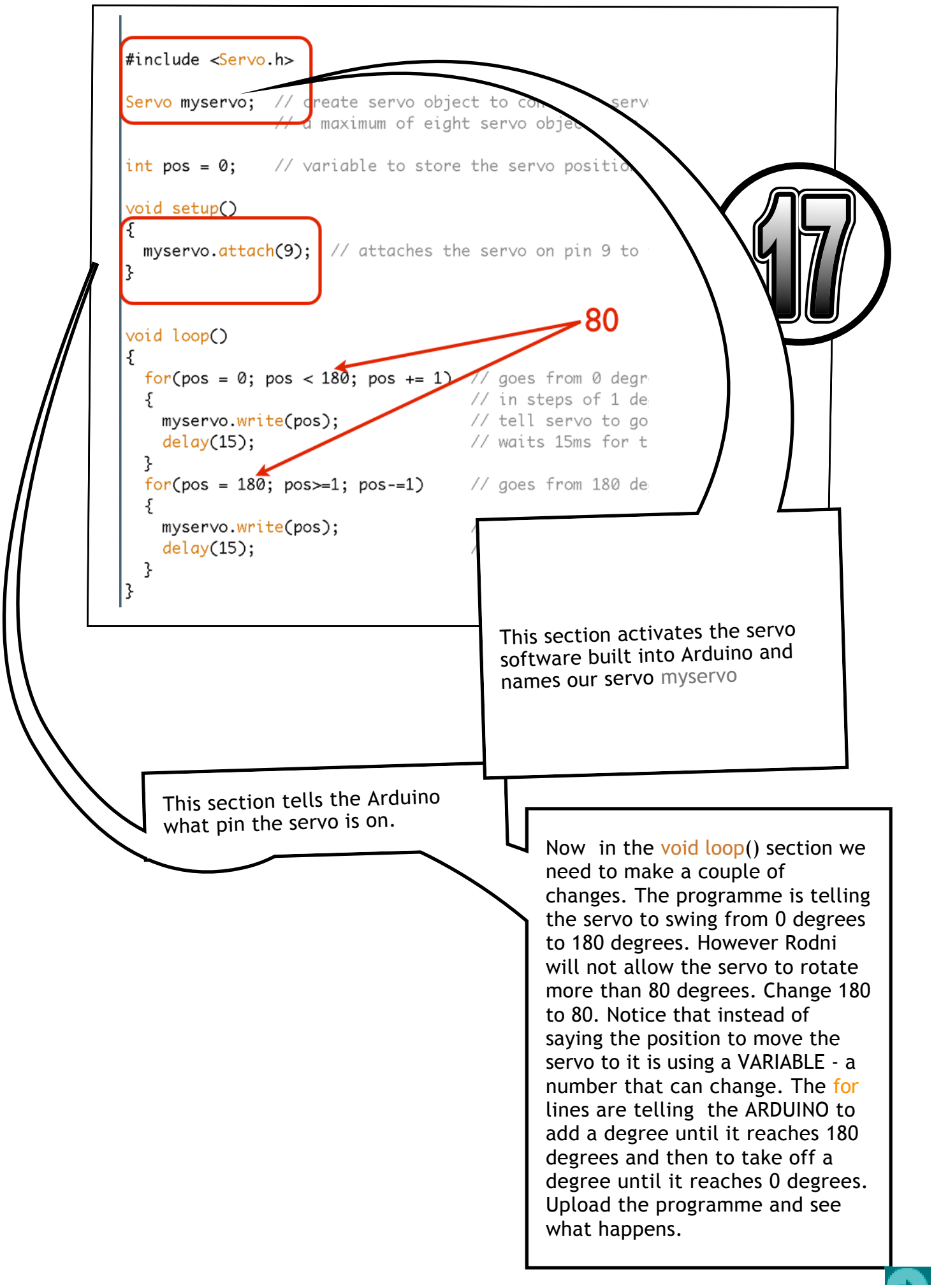
15

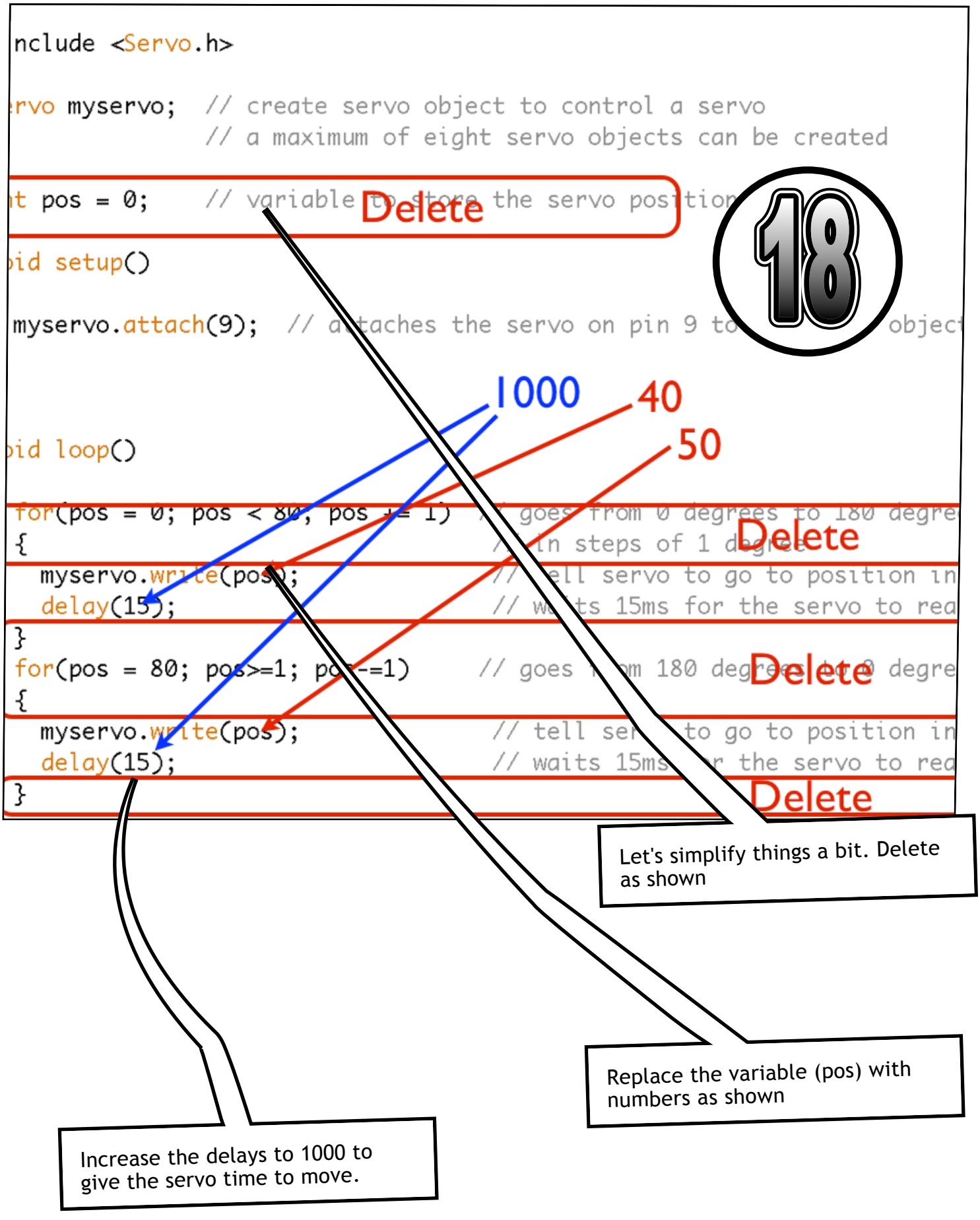




16

So far so good. Now let's look at the software needed to run the servo.
This is surprisingly easy to set up. We will simply copy it from an example.
Go to the Arduino programme File>Examples>Servo>Sweep and open the Sweep programme. This will open in a new window.





```
nclude <Servo.h>

rvo myservo;  // create servo object to control a servo
              // a maximum of eight servo objects can be creat



id setup()

myservo.attach(9);  // attaches the servo on pin
```

19

Try changing these numbers to between 0 an

```
id loop()


  myservo.write(40);              // tell servo to go to posit
elay(1000);
  myservo.write(60);
  delay(1000);
```

TRY KEEPING BOTH NUMBERS THE SAME
TRY MAKING THE NUMBERS DIFFERENT

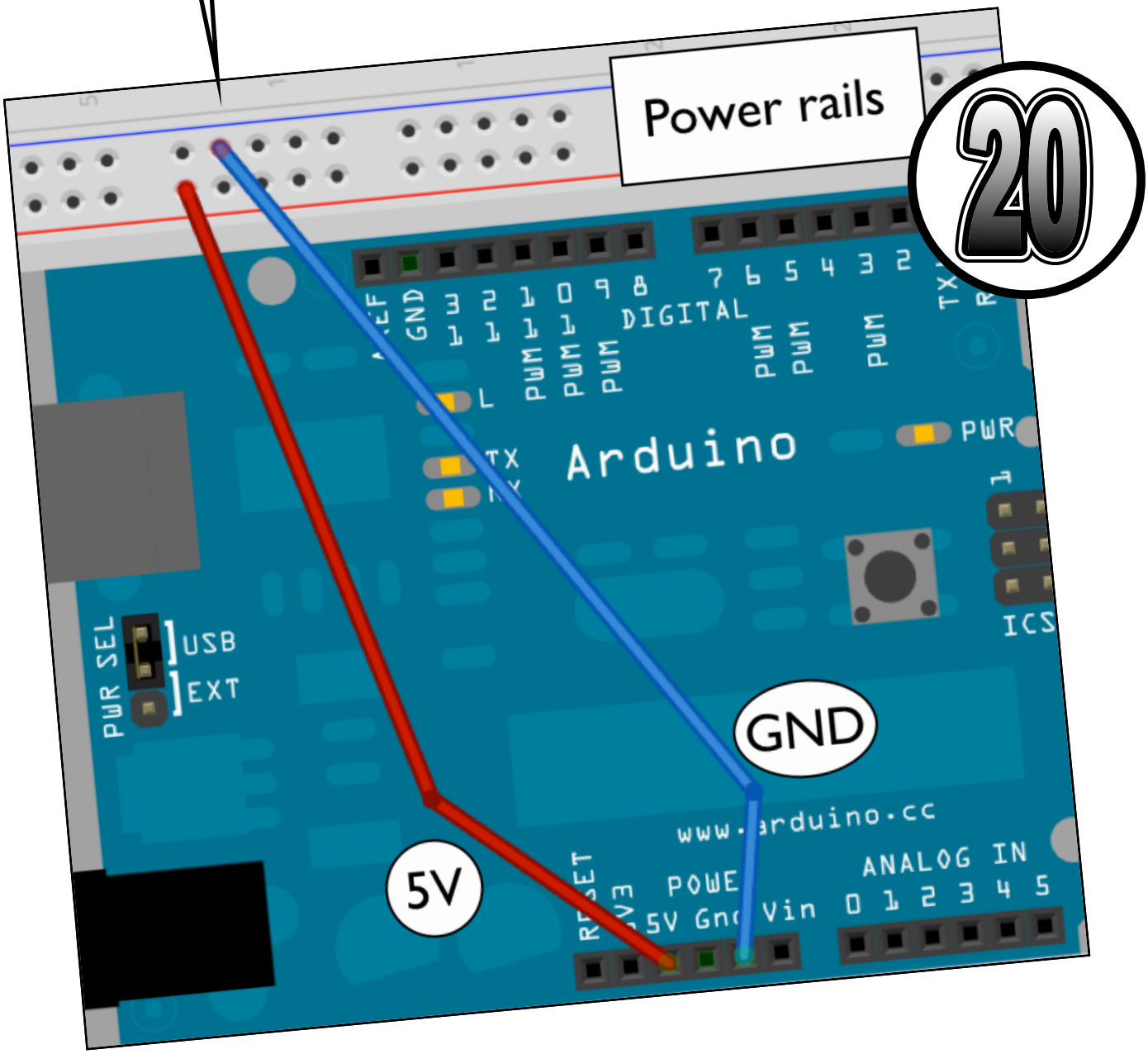
TRY 40 AND 60

TRY CHANGING THE DELAYS (5000 FOR INSTANCE)

Well done. Let's save this programme as Rodniservo. Then let's celebrate by finishing building Rodni. Choose a tail and other parts that you want to use to customise him.

Making Rodni see

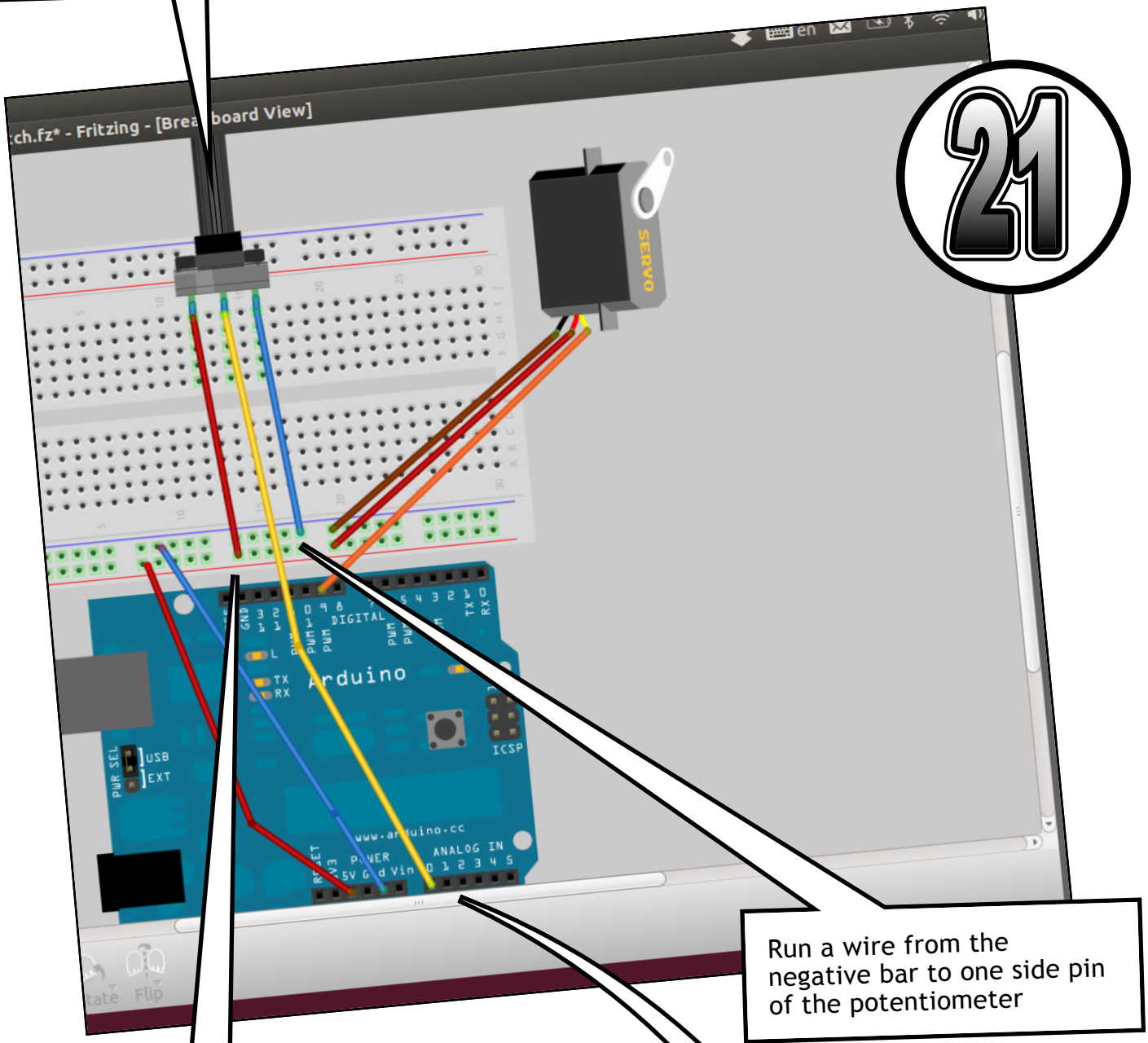
Set up the power rails on the breadboard



Making Rodni see

21

Carefully push a potentiometer into the breadboard

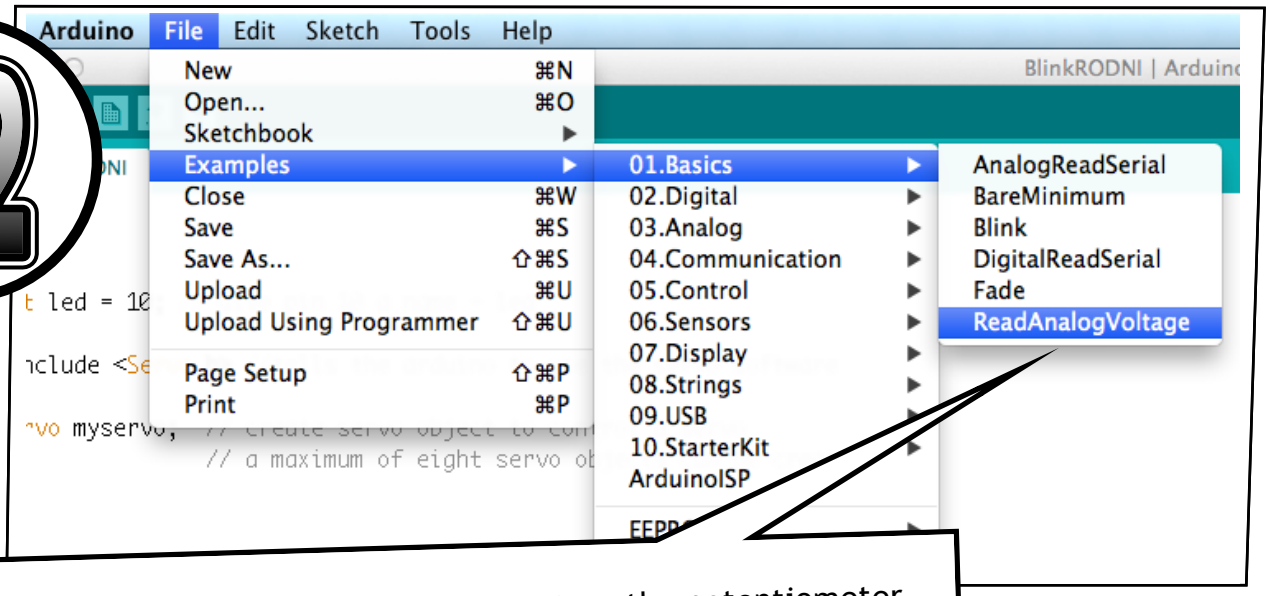


Run a wire from the negative bar to one side pin of the potentiometer

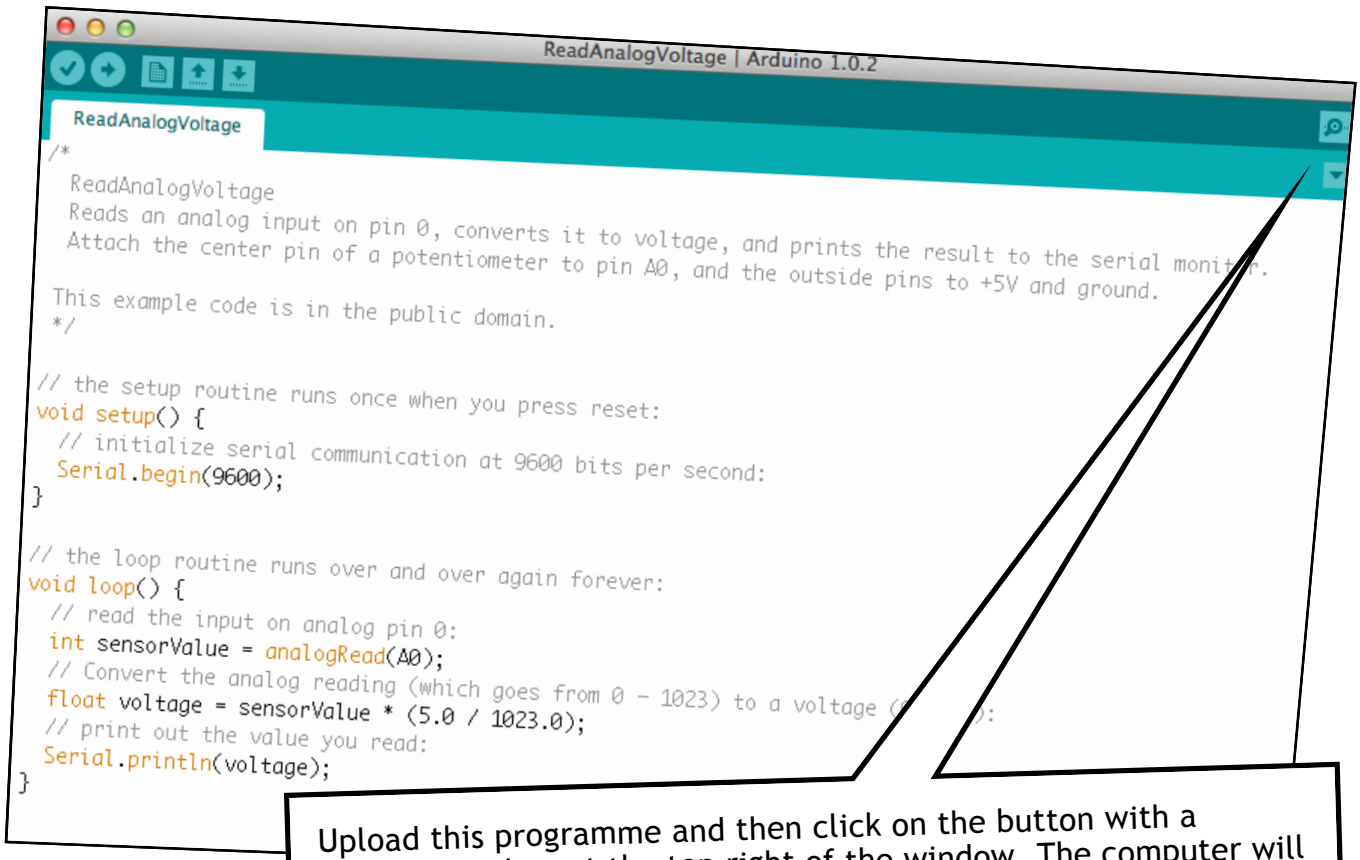
Run a wire from the positive bar to one side pin of the potentiometer

Finally run a wire from the central pin on the potentiometer to the socket A0 on the Arduino.

22



We will now see what happens when we turn the potentiometer knob. Once again, it is very easy with Arduino because a basic programme has been written for us and can be found in Examples. Open File>Examples>01 Basics>ReadAnalogVoltage.



Upload this programme and then click on the button with a magnifying glass at the top right of the window. The computer will start talking to the Arduino and will show the voltage it is reading as commanded in Serial.println(voltage);

23

Turn the potentiometer knob and see what happens to the numbers

24

Load ReadAnalogVoltage again

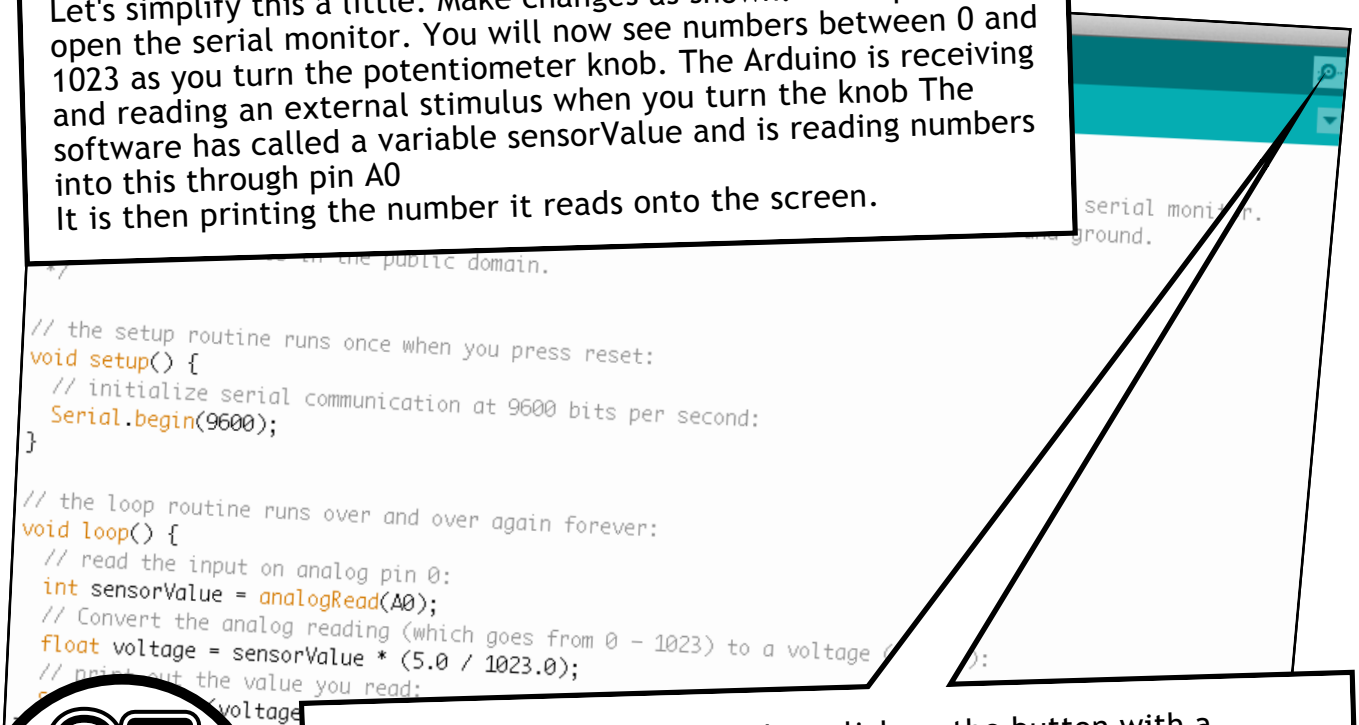
```
etup() {
 initialize serial communication at 9600 bits per second:
Serial.begin(9600);

// the loop routine runs over and over again forever:
oid loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0);
 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 float voltage = sensorValue * (5.0 / 1023.0);
 // print out the value you read:
 Serial.println(voltage);
}
```

Delete this line

Change to (sensorValue);

We can now read number between 0 and 1023

Let's simplify this a little. Make changes as shown. Now upload and open the serial monitor. You will now see numbers between 0 and 1023 as you turn the potentiometer knob. The Arduino is receiving and reading an external stimulus when you turn the knob The software has called a variable sensorValue and is reading numbers into this through pin A0
It is then printing the number it reads onto the screen.



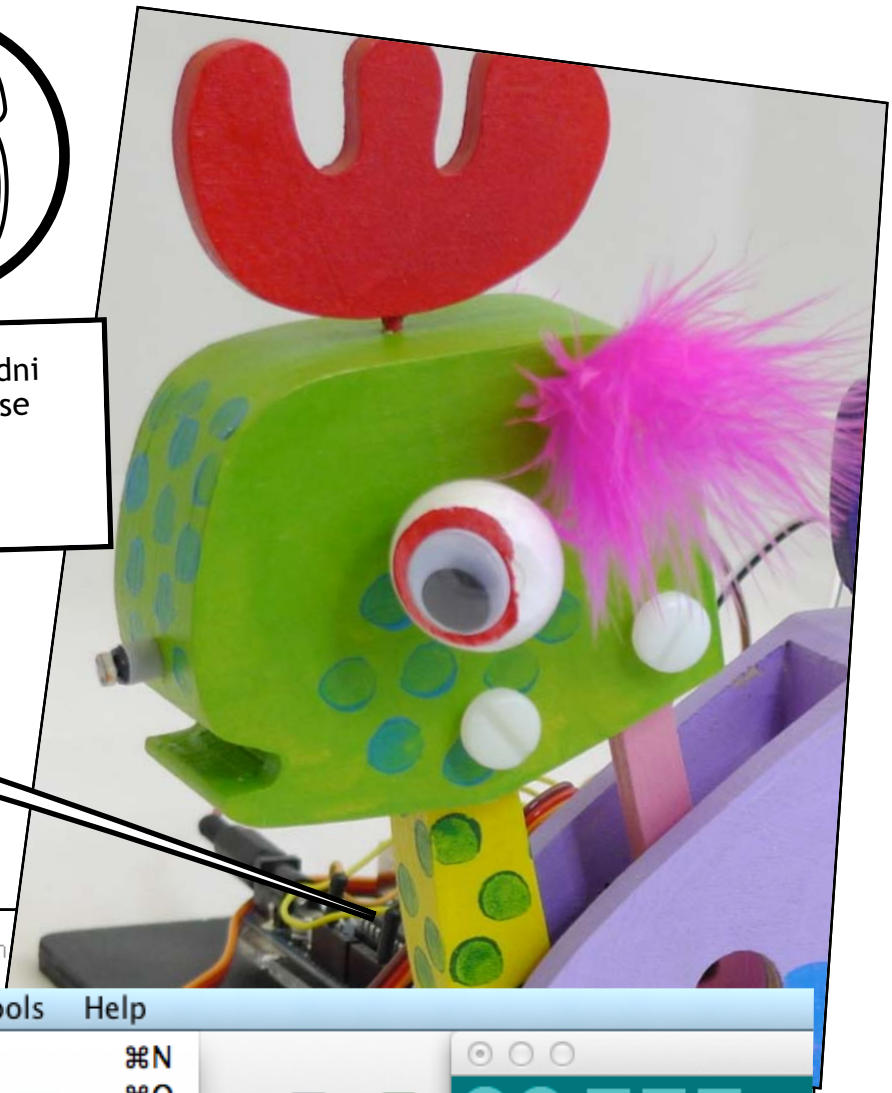
25

Upload this programme and then click on the button with a magnifying glass at the top right of the window. The computer will start talking to the Arduino and will show the voltage it is reading as commanded in Serial.println(voltage);

Turn the potentiometer and see what happens to the numbers

26

Let's use this now to make Rodni come alive. We are going to use the variable from the potentiometer to control the servo



Load the Knob programme.

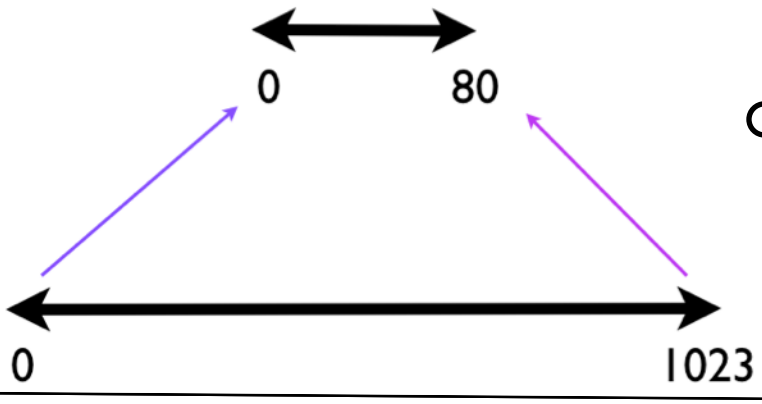
27

```
// by Michal Rinott <http://people.interaction-ivrea

#include <Servo.h>

Servo myservo;  // create servo object to control

int potpin = 0;  // analog pin used to connect the pot
int val;    // variable to read the value from the analog pin

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo obj
}

void loop()
{
  val = analogRead(potpin);            // reads the value of the potenti
  val = map(val, 0, 1023, 0, 179);     // scale it to use it with the ser
  myservo.write(val);                  // sets the servo position accordi
  delay(15);                           // waits for the servo to get there
}
```

Change to 80);

This line is mapping the range 0 to 1023 to 0 to 179. (So 1023 becomes 179). However Rodni can only handle the servo turning up to 80 degrees so change 179 to 80.
Now changes in the variable coming from the potentiometer will be converted from a range of 1 to 1023 to a range of 0 to 80 degrees.

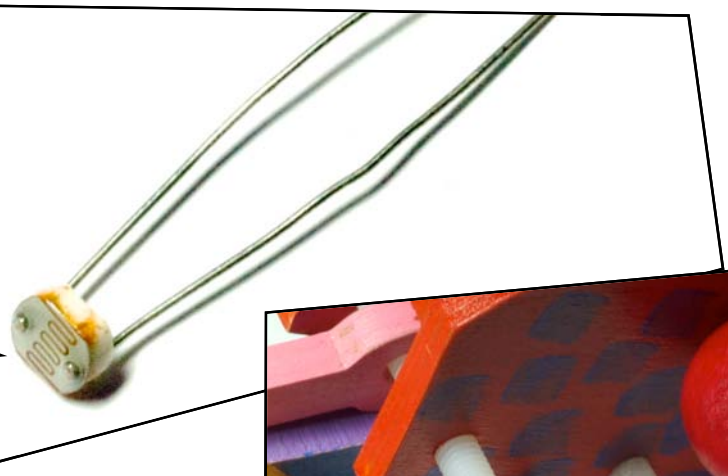
map

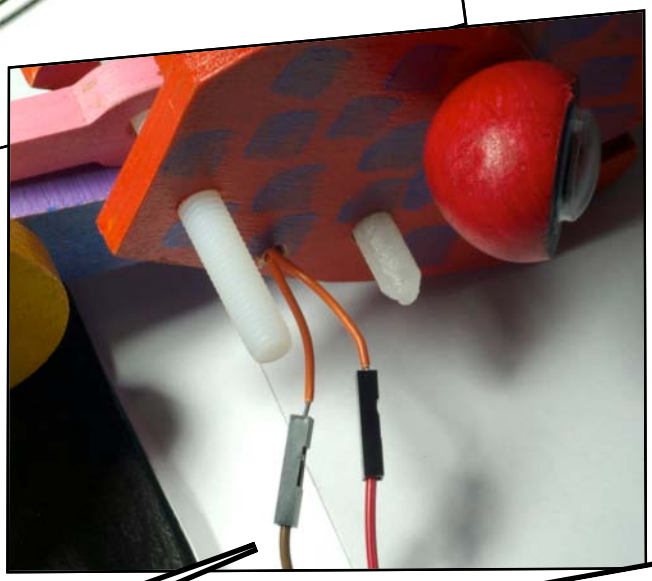


SAVE AS RODNIMOVE

TURN THE POTENTIOMETER KNOB AFTER UPLOADING

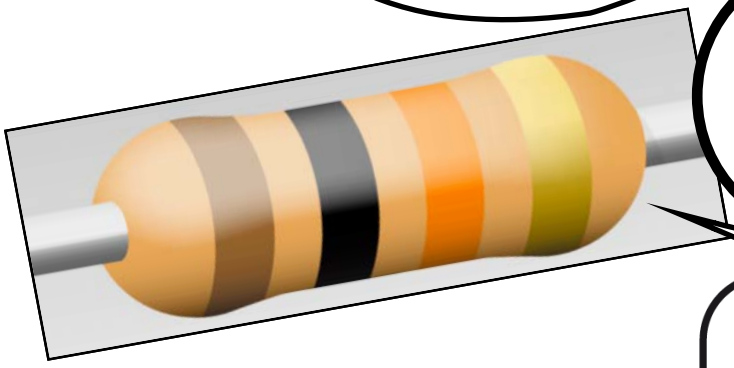
Inside Rodni's head there is a small electronic part called a light sensitive resistor. In a bright light this component has a low resistance and in a dim light it has a high resistance. We can use this to measure the light intensity and give Rodni "sight".





28

Take 2 wires with sockets at one end and pins at the other and attach the sockets to the wires coming out of Rodni's head.



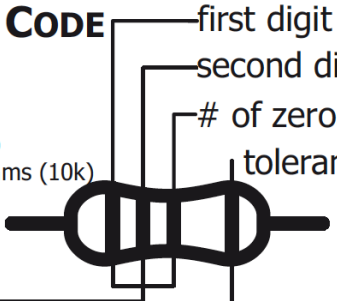
29

Now find a 10,000 ohm resistor. Check with the colour code chart.

Resistor Color Code

Examples:

green-blue-brown - 560 ohms
red-red-red - 2 200 ohms (2.2k)
brown-black-orange - 10 000 ohms (10k)



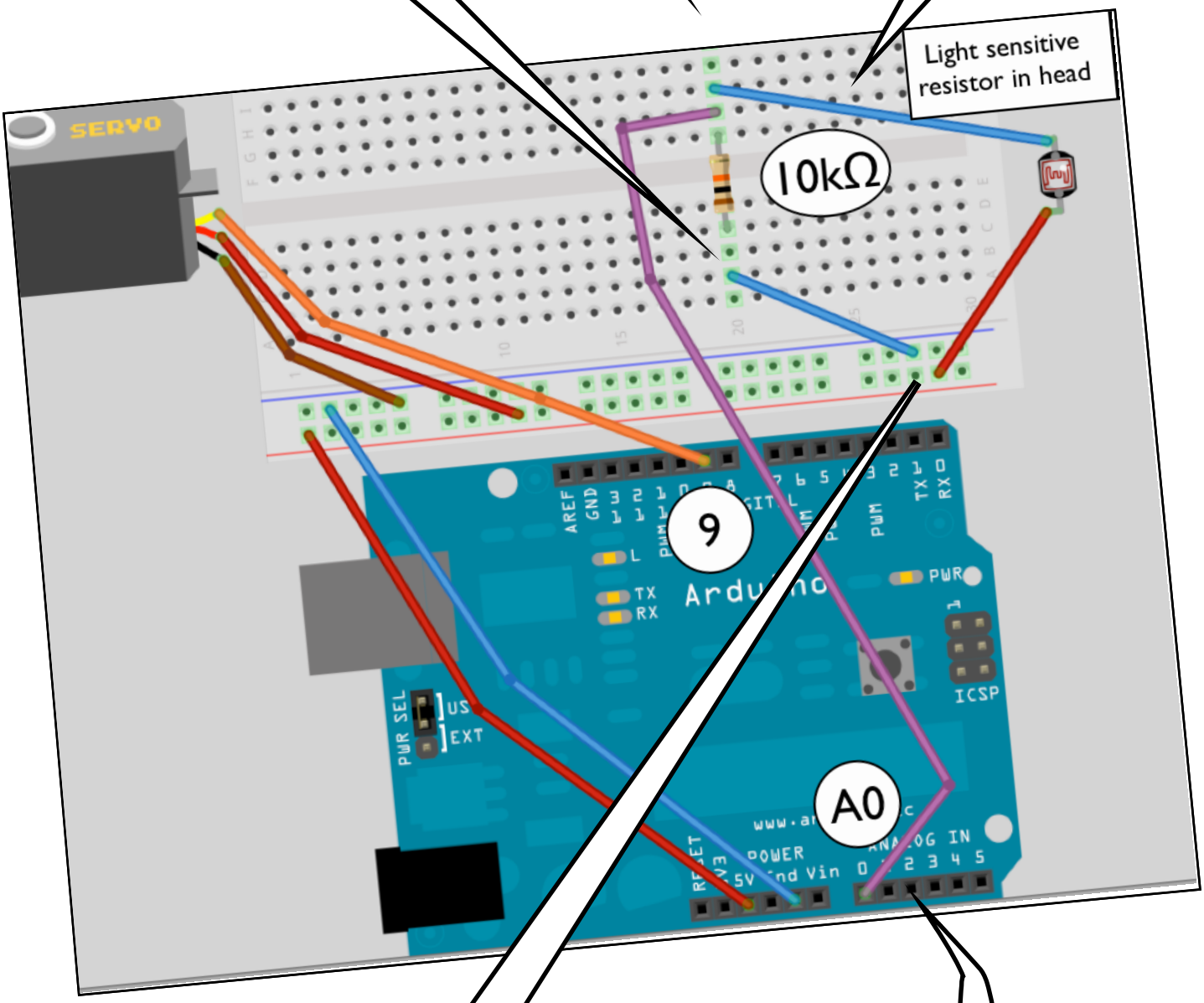
0 - Black	5 - Green	20% - none
1 - Brown	6 - Blue	10% - silver
2 - Red	7 - Purple	5% - gold
3 - Orange	8 - Grey	
4 - Yellow	9 - White	

30

Now we will replace the potentiometer with a resistor and a light sensitive resistor

Connect the other side of the resistor to the negative.

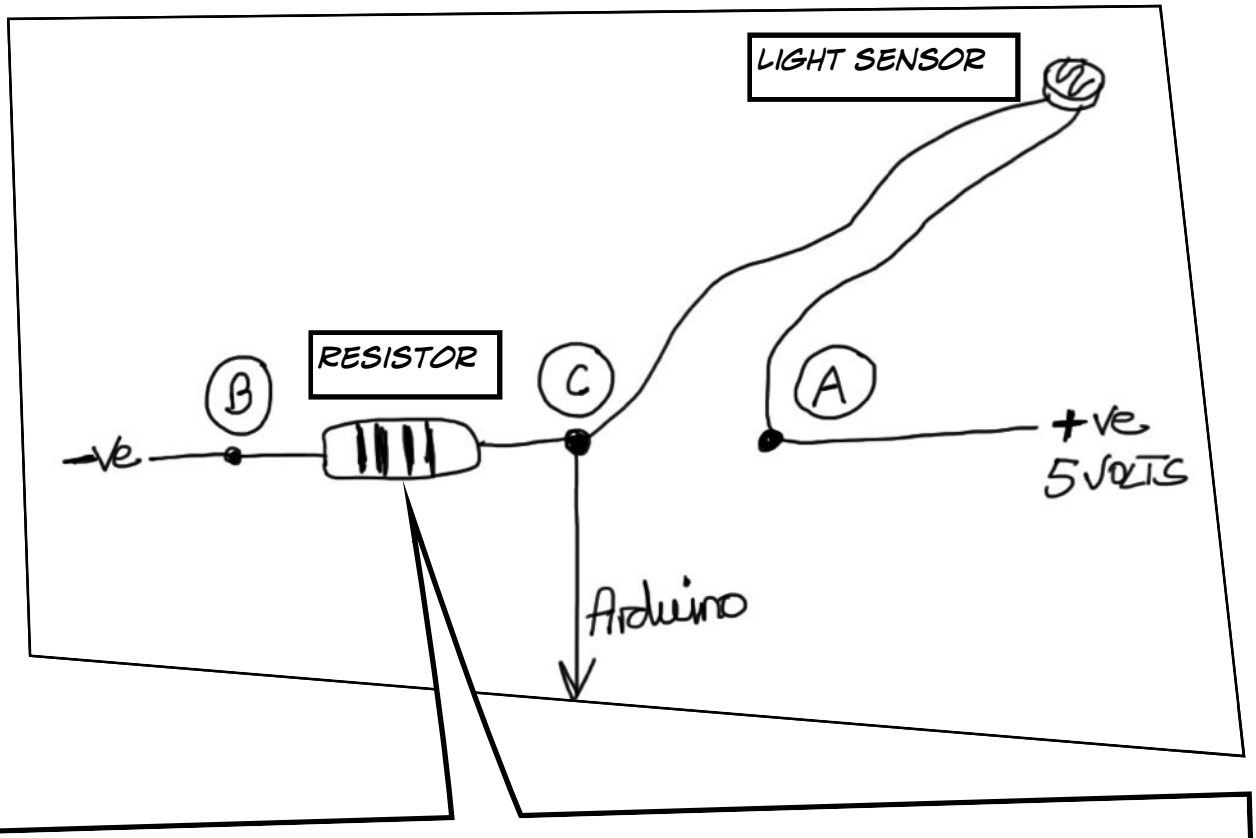
Insert one wire from the light sensitive resistor on one side of the resistor



Connect the other wire from the light sensitive resistor to the +ve (red line) bar.

Finally run a wire from the resistor to the socket A0 on the Arduino.

DON'T WORRY TOO MUCH ABOUT THIS PAGE. YOU CAN MISS IT OUT IF YOU LIKE BUT YOU MIGHT LIKE TO KNOW A LITTLE ABOUT WHAT WE ARE TRYING TO DO.



Let's take a look at what we just did.
We are trying to measure the amount of light hitting the light sensitive resistor.
This light affects its resistance.
The problem that we have to overcome is that the Arduino cannot measure resistance directly. However it can measure voltage. If electricity was water, voltage would be the water pressure.
Our circuits are running on 5 volts so there is 5 volts difference between +ve and -ve. (A) and (B). We need not worry about the detail here but if we have 2 resistors, the 10,000 ohm one and the light sensitive resistor we can measure the difference in voltage between them.
The sockets beginning with A on the Arduino are analog pins. The other ones are digital pins. Digital pins only read and write on and off. Analog pins can tell the difference between different levels of voltage.
The Arduino converts the voltage to a number between 0 and 1023.
We will try this out next to see how it works.

31

Load ReadAnalogVoltage again

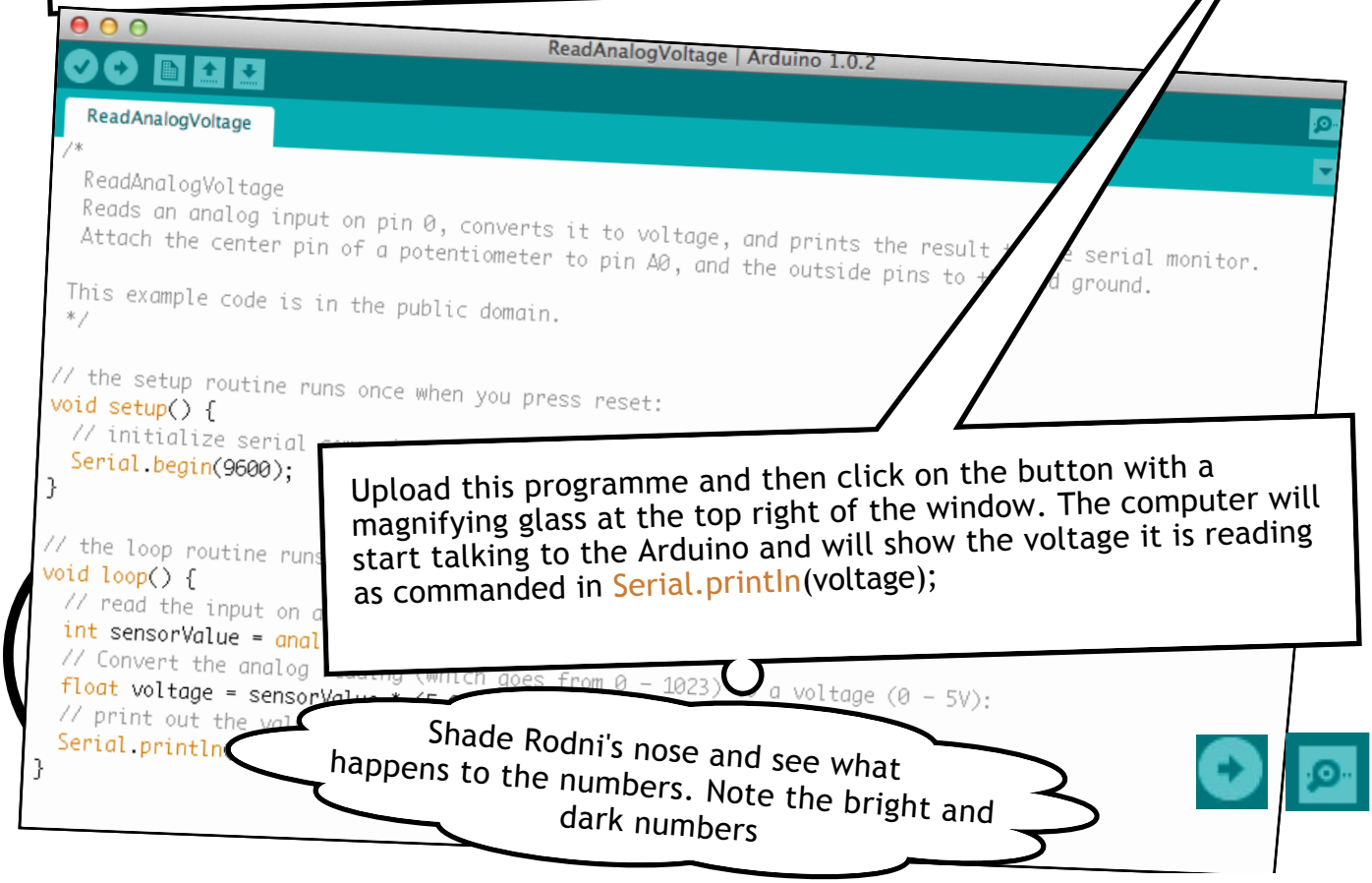
```
etup() {
 initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Delete this line

Change to (sensorValue);

We can now read number between 0 and 1023

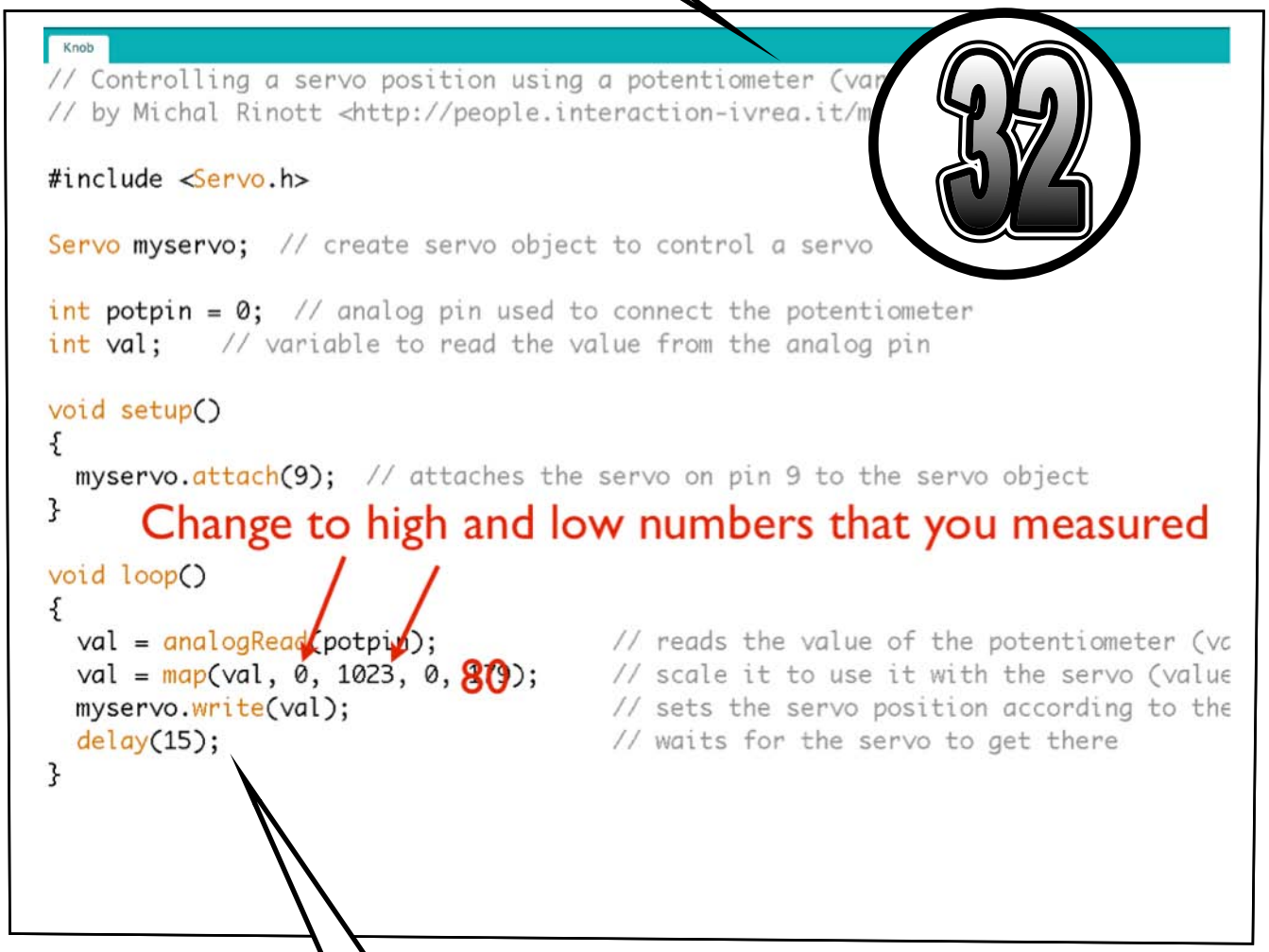
Let's simplify this a little. Make changes as shown. Now upload and open the serial monitor. You will now see numbers between 0 and 1023 as you turn the potentiometer knob. The Arduino is receiving and reading an external stimulus when you turn the knob The software has called a variable sensorValue and is reading numbers into this through pin A0
It is then printing the number it reads onto the screen.



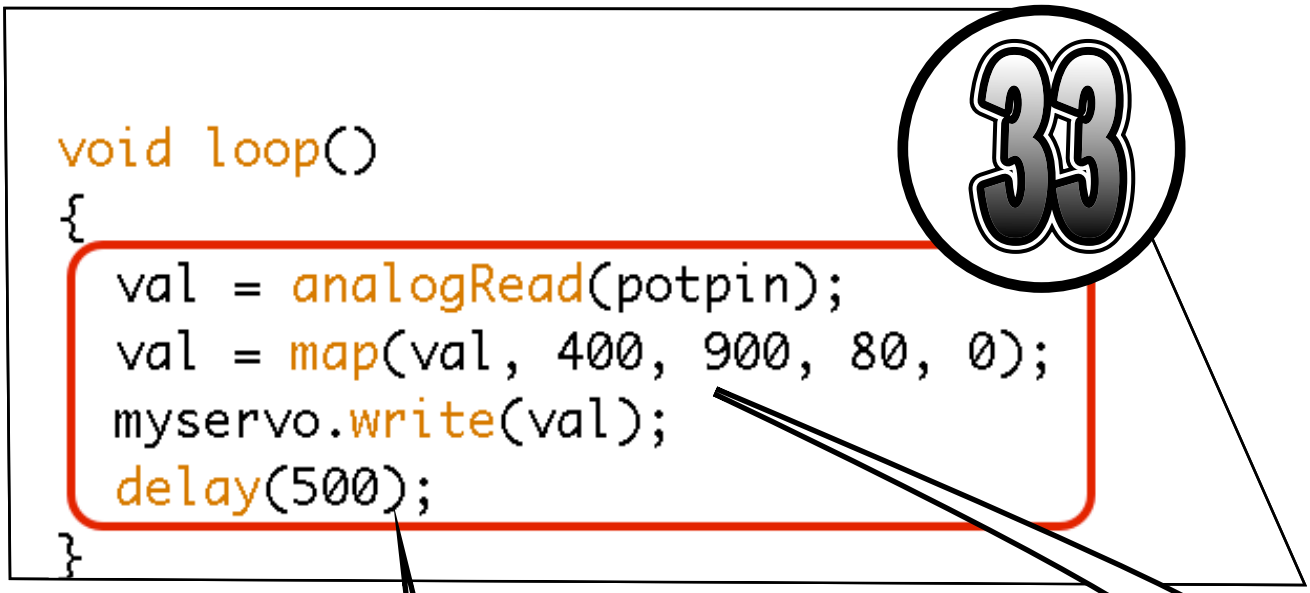
Upload this programme and then click on the button with a magnifying glass at the top right of the window. The computer will start talking to the Arduino and will show the voltage it is reading as commanded in Serial.println(voltage);

Shade Rodni's nose and see what happens to the numbers. Note the bright and dark numbers

Reload the programme you saved as Rodnisee.
Alternatively open Knob again.



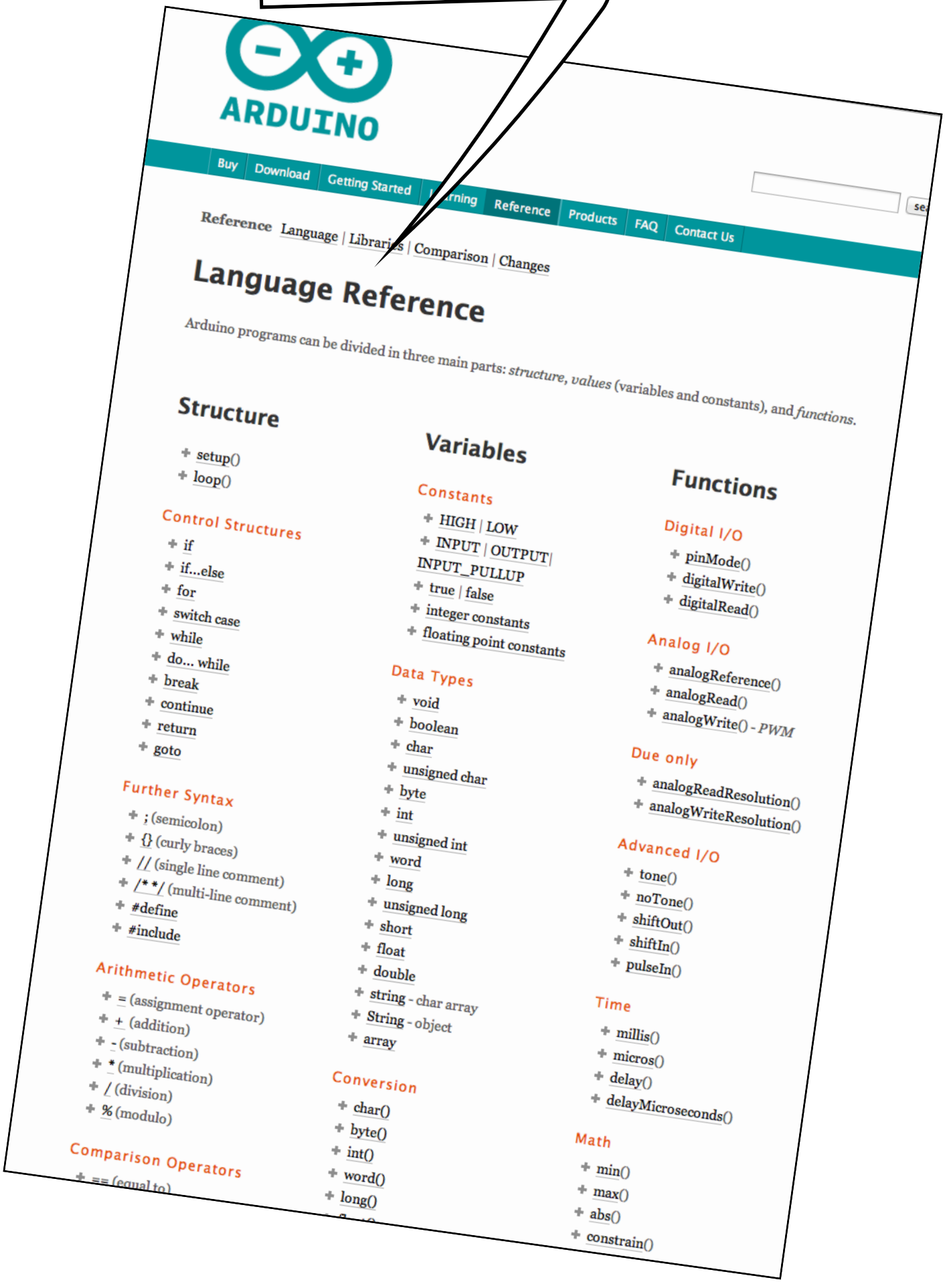
Try varying the delay time and also the map numbers.
Experiment.



When we measured the variable coming front the light sensor we got a narrower ranger than 0 to 1023. Substitute the numbers that you recorded and then upload.

Try varying the delay time and also the map numbers. Experiment.

All the commands and how to use them can be found on the arduino.cc website



ARDUINO

Buy | Download | Getting Started | Learning | Reference | Products | FAQ | Contact Us

Reference Language | Libraries | Comparison | Changes

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string - char array
- String - object
- array

Conversion

- char()
- byte()
- int()
- word()
- long()

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()

CHALLENGE

CAN YOU NOW ADD HIS HEART BEAT?

CHALLENGE

AT THE MOMENT YOU HAVE RODNI REACTING TO DIFFERENT LIGHT LEVELS. IT WOULD BE BETTER FOR INTERACTIVITY IF HE REACTED TO A CHANGE IN LIGHT LEVEL RATHER THAN THE ABSOLUTE LEVEL.
HOW WOULD YOU DO THIS?

CHALLENGE

HOW WOULD YOU MAKE RODNI GO TO SLEEP WHEN IT GETS DARK?

TOUGHER CHALLENGE

HOW WOULD YOU MAKE RODNI GO TO SLEEP WHEN HE GETS BORED?