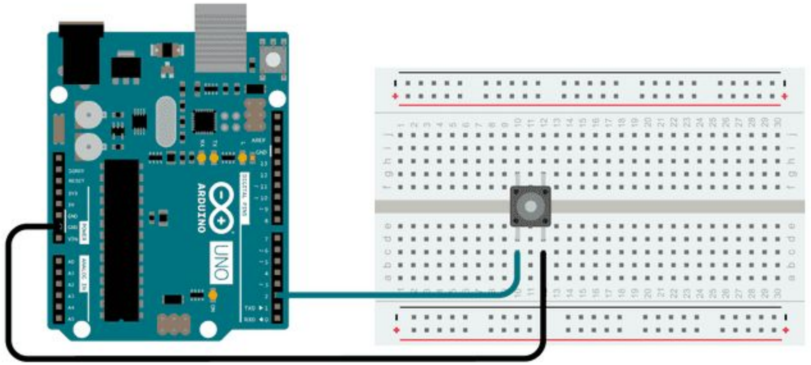
Electronics

Programming 102

Programming 102 >> Pullup

When using a push button switch, we can use the pinMode(pin, INPUT_PULLUP) to send a current through the pin.

That way it reads HIGH when the button is neutral, and LOW when the button is pressed, because the current is then pulled through the switch to the GND.



Programming 102 >> Variables

Variables are used to store data that can change (as opposed to constants, which contain the same information for the whole programme).

We'll use variables to hold the data being input to the programme, for example the values read by the sensors.

Programming 102 >> Variables

When we declare the variable we can give in an initial value or leave it empty until it receives data.

```
int photo_input = 513;
```

Or just:

```
int photo_input;
```

Programming 102 >> Variables

In most cases we'll insert data into the variables using either digitalRead() or analogRead(). The syntax is simple: digitalRead(pin), or analogRead(pin)

For example:

photo_input = analogRead(3);

What this does is store whatever value is read from pin number 3 into the variable named photo_input. If there was a value there before it will replace it.

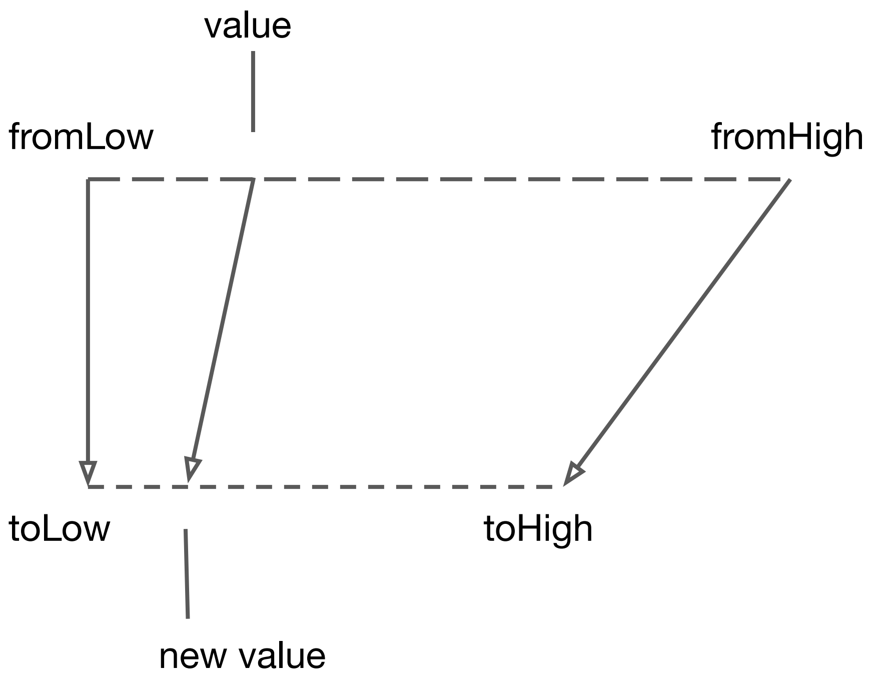
```
int photo_input = 0;

void setup() {
}

void loop() {
  // put your main code here, to run repeatedly:
photo_input = analogRead(3);
```

Programming 102 >> Variables >> Map

map() is used to remap numbers from one range to another.
The syntax is:

map(value, fromLow, fromHigh, toLow, toHigh)

This is most often used to remap analog values (that range from 0 to 1023) to another range that is useful for our programme.
This could be brightness of a light, speed of a motor etc



Programming 102 >> Variables >> Map

For example, the following code maps the analog input from a potentiometer to the brightness value of an LED.

First the value from the potentiometer is assigned to the variable poten_val.
Then the variable brightness gets assigned the returned value of the map() calculation.
In this case our new range starts from 10 so that the LED is never completely off.

```
const int LEDpin =3;
int brightness;
int poten_val;

void setup() {
  pinMode(LEDpin, OUTPUT);
}

void loop() {
poten_val = analogRead(A0);
brightness = map(poten_val, 0, 1023, 10, 255);
analogWrite(LEDpin, brightness);
delay(1000);
}
```

Programming 102 >> If

An if statement is used to run parts of the code only if some condition holds.

Conditions are statements that are evaluated to boolean values. If the statement is TRUE then the code will run, otherwise it will be skipped. The syntax is:

```
if (condition) {
  //statement(s)
}
```

Programming 102 >> Comparison operators

In different loops or statements
we will test conditions in order
to execute some parts of code,
but not others.
The result of a condition is a
boolean value, either TRUE or
FALSE.
If the condition is TRUE, the
following block of code will be
executed.
The operators are:
x == y (x is equal to y)
x != y (x is not equal to y)
x < y (x is less than y)
x > y (x is greater than y)
x <= y (x is less than or equal to y)
x >= y (x is greater than or equal to y)

Programming 102 >> Comparison operators

Note that in evaluating a statement we use a double equal sign:

```
If (x == y){
}
```

The single equal sign is used in coding to assign values to variables.
For example when declaring a variable:

```
int photo_input = 513;
```

Programming 102 >> Comparison operators

We can also test for more than one condition.

&& means and

|| means or

In the first condition, the LED will only turn on if both buttons are pressed.

In the second condition, the LED can be turned on by pressing either button.

```
void loop() {

if(button_1 == HIGH && button_2 == HIGH){
  digitalWrite(LED_pin, HIGH);
}

if(button_1 == HIGH || button_2 == HIGH){
  digitalWrite(LED_pin, HIGH);
}
}
```

Programming 102 >> If >> Else

The else statement is used to execute an alternative part of code when the if condition doesn’t hold.

This means that only one block of code will be executed, but never both.

```
if (temperature >= 70) {
  Serial.print("Danger! Shut down the system.");
}

    else { // temperature < 70
      Serial.print("Safe! Continue usual tasks.");
  }
```

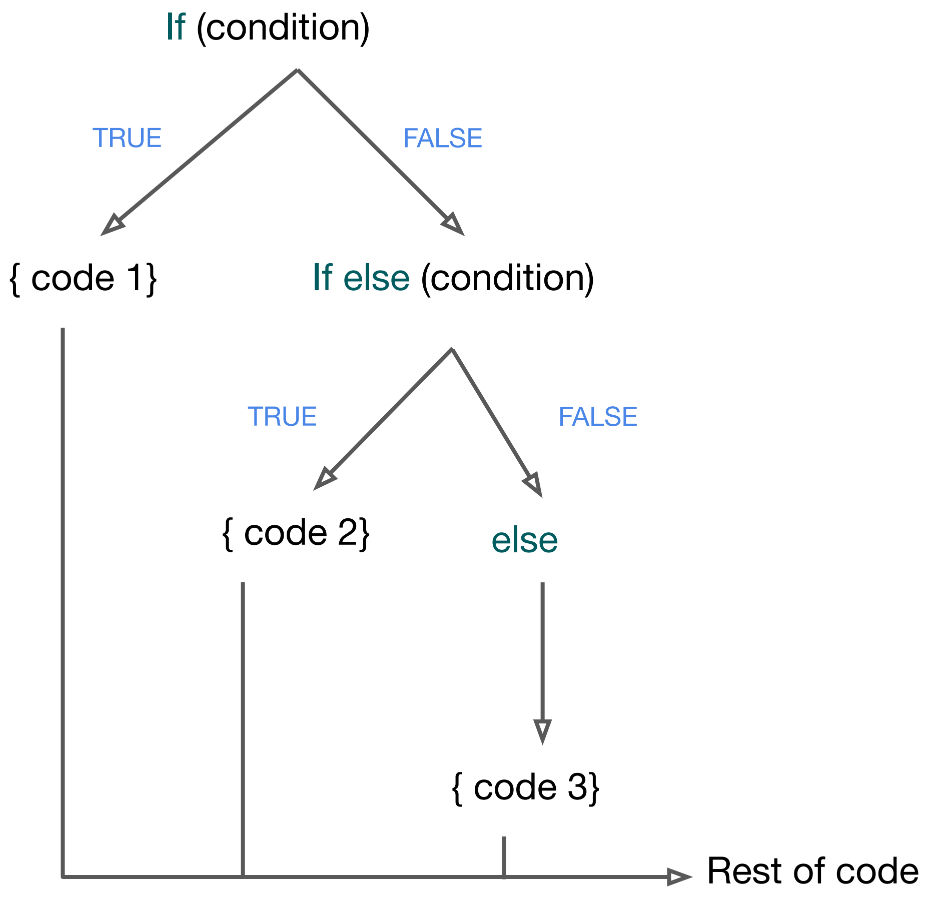
Programming 102 >> If >> Else

else if can be used to test another condition and only execute something if that holds instead.

In this case, each step will only be checked if the previous conditions weren't met.

```
if (temperature >= 70) {
  Serial.print("Danger! Shut down the system.");
}
  else if (temperature >= 60) { // 60 <= temp < 70
    Serial.print("Warning! User attention required.");
  }
    else { // temperature < 60
      Serial.print("Safe! Continue usual tasks.");
  }
```

Programming 102 >> If >> Else



```
if (temperature >= 70) {
  Serial.print("Danger! Shut down the system.");
}
  else if (temperature >= 60) { // 60 <= temp < 70
    Serial.print("Warning! User attention required.");
  }
    else { // temperature < 60
      Serial.print("Safe! Continue usual tasks.");
  }
```

Programming 102 >> Loops

Loops are used to execute portions of code again and again, a set number of times or until a certain condition is met.

The main portion of the arduino porgramme is a loop which doesn’t have a condition, so it runs forever.

We can also add loops into our code to repeat portions of code or do something until a parameter changes.

Programming 102 >> Loops >> While

A while loop will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. The syntax is:

```
while (condition) {
  // statement(s)
}
```

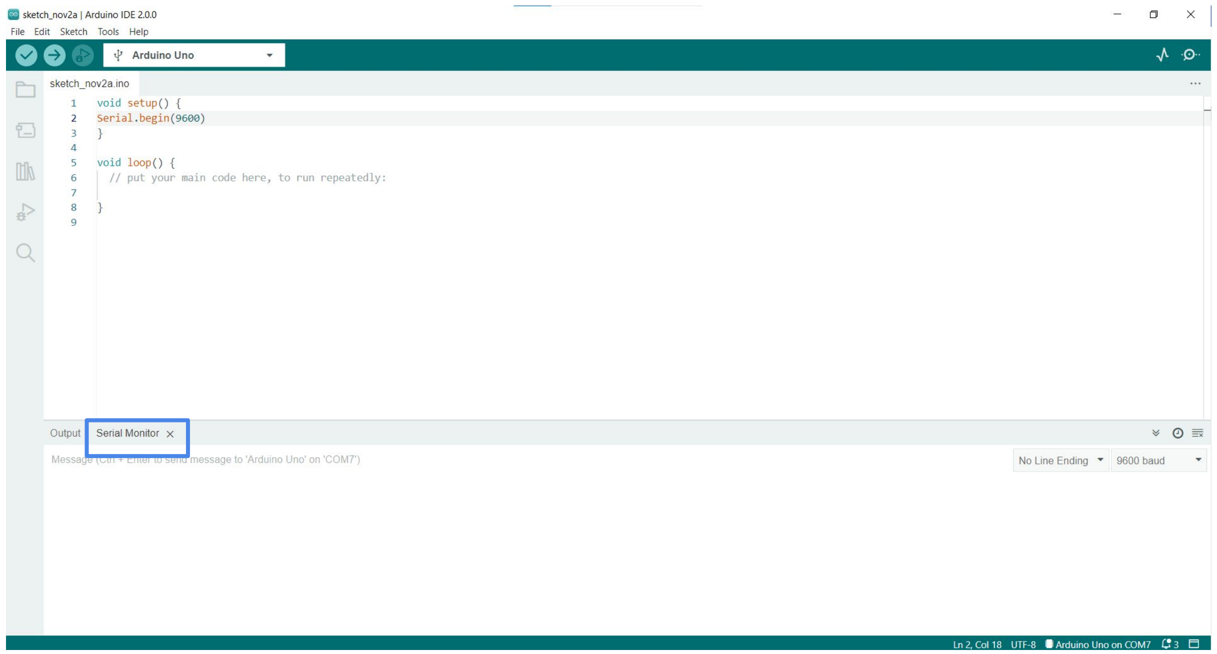
Something must change the tested variable, or the while loop will never exit. This could part of the code, such as an incremented variable, or an external condition, such as testing a sensor.

Programming 102 >> Serial communication

The serial monitor is used by arduino to communicate data with the computer.

We can use it to display information from the code, or to input information to the code.

The serial monitor can be found at the bottom of the IDE.



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

void loop() {
  // put your main code here, to run repeatedly:

}
```

Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM7')

No Line Ending 9600 baud

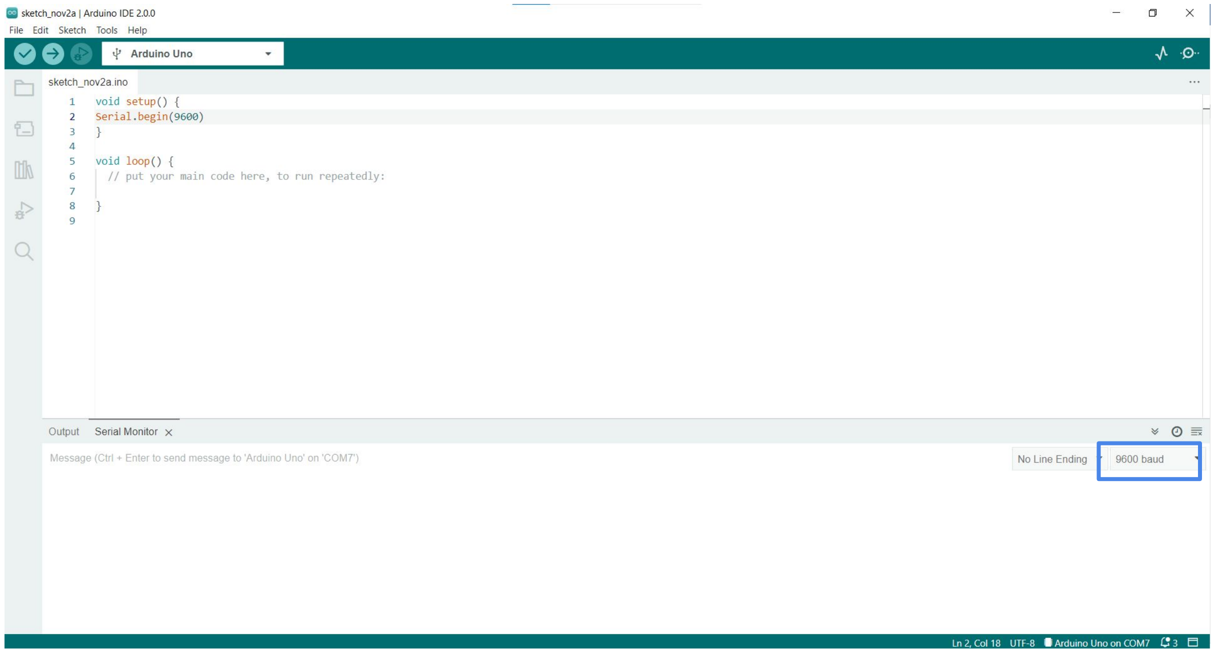
Ln 2, Col 18 UTF-8 Arduino Uno on COM7 3

Programming 102 >> Serial communication

In order to use the the serial monitor we need to enable it. We do this using the following function under the setup() section, because this function needs to only run once:

```
Serial.begin(9600);
```

The number 9600 is the baudrate, or the speed that the monitor refreshes. We'll use 9600 and make sure the monitor is on 9600 as well.


sketch_nov2a | Arduino IDE 2.0.0
File Edit Sketch Tools Help
Arduino Uno
sketch_nov2a.ino
Output
Serial Monitor
Ln 2, Col 18 UTF-8 Arduino Uno on COM7

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

void loop() {
  // put your main code here, to run repeatedly:

}
```

Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM7')

No Line Ending

9600 baud

Programming 102 >> Serial communication >> Print data

In order to display a message or data to the user, we can print information to the monitor. This can be done using

Serial.print()
Or
Serial.println()

Both print the value directly next to the previous content, and we'll use the second if we want to start a new line directly after.

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

void loop() {
  // put your main code here, to run repeatedly:
Serial.print("Message here: ");
Serial.println("message");

delay(1000);
}
```

Serial Monitor:

```
Message here: message
Message here: message
```

Programming 102 >> Serial communication >> Print data

If we want to add a new line in a specific location, we can use the character “\n”.

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

void loop() {
  // put your main code here, to run repeatedly:
Serial.print("\n" "Message here: ");
Serial.print("message");

delay(1000);
}
```

Serial Monitor:

```
Message here: message
Message here: message
Message here: message
```

Programming 102 >> Serial communication >> Print data

“ “ are used to contain string, or text.

We can also print a integer directly, or print the value of a variable by calling the variable name.

```
int num = 5;

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

void loop() {
  // put your main code here, to run repeatedly:
Serial.print("The number is: ");
Serial.println(num);

delay(1000);
}
```

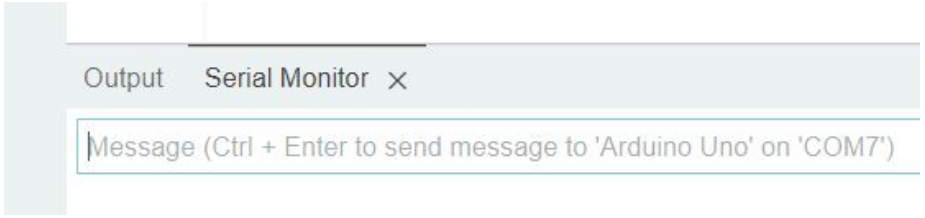
Serial Monitor:

```
The number is: 5
The number is: 5
The number is: 5
```

Programming 102 >> Serial communication >> Read data

We can also use the serial monitor to have the user input data to the code.

We do that by putting our data into the message line.



Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM7')

Programming 102 >> Serial communication >> Read data

We then tell the code to expect data and then read it.
We will use a different function to read the data depending on the type of data we are inputting:

Serial.parseInt() for numbers
or
Serial.readString() for text

Programming 102 >> Serial communication >> Read data

In order for the programme to wait until an input was sent through the monitor, we'll use a while loop.

```
while (Serial.available() == 0){
}
```

The Serial.available() checks if there is data in the command line. If it's empty, it will return zero and therefore the loop will execute the empty brackets and immediately check again. Once there is any data entered, it no longer equals zero and the code will continue past the loop.

Programming 102 >> Serial communication >> Read data

Once we receive data from the serial monitor we need to store it in a variable in order to use it in the rest of our code.
Let's say we declared a variable:

```
String userInput;
```

We will store the data in it like so:

```
userInput = Serial.readString();
```

Now the variable userInput will contain whatever message the user entered.