CPSC 1000: Introduction to Computer Science

Arduino basic program template

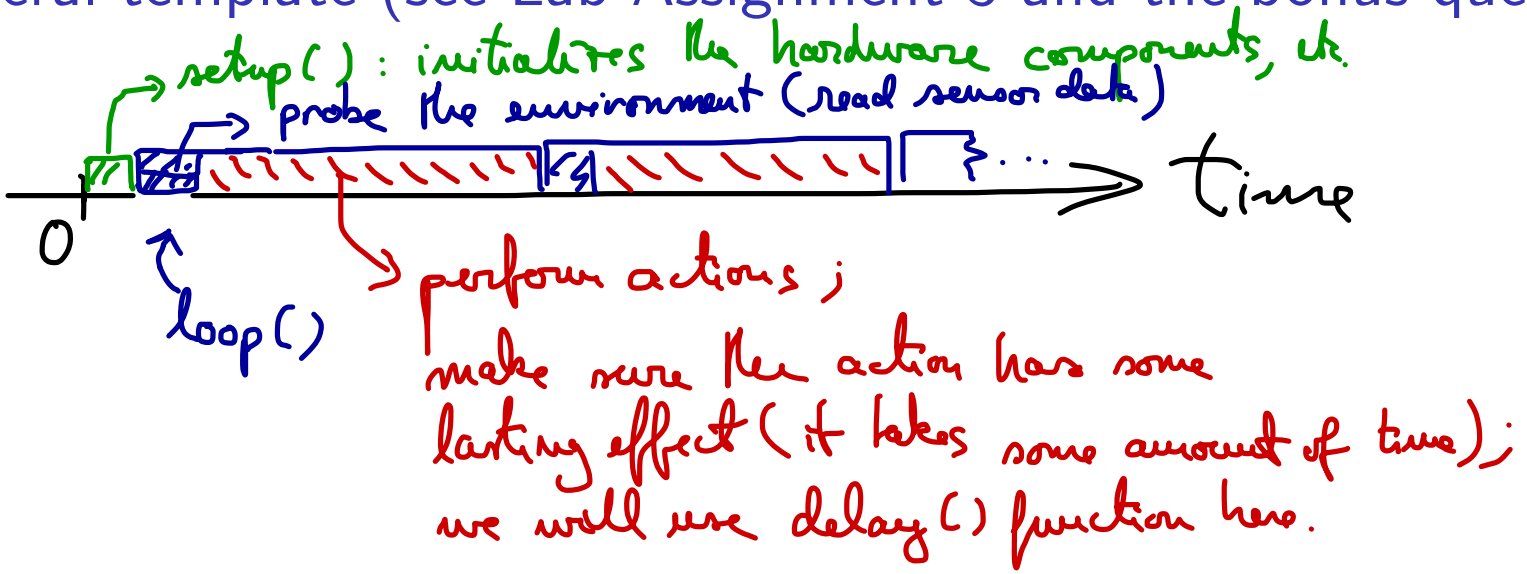
Robert Benkoczi, C556
robert.benkoczi@uleth.ca

(9,11)-Oct-2018
(Week 5)

Objectives

- Using the Arduino Notebook and the Arduino Programming Language Reference, both available on Moodle, and the functions explained here, students will write Arduino programs that read input from sensors or the serial monitor, and perform actions based on these inputs.
- Students will write Arduino programs that receive data to/from the serial monitor.
- Students will use `random` and `randomSeed` functions to add "chance" to their programs.
- Students will write loop statements.

The general template (see Lab Assignment 3 and the bonus question)



Code example for the template:

```
void setup() { ... }
  // declare global variables to encode the state of the environment & device
  void loop() {
      ⋮ } probe environment   (set the state variables)

      ⋮ } generate action (signals)   (use the state variables)

  }
```

Sending data to the serial monitor (recap)

- `Serial.begin(9600);`

- `Serial.println(‘‘Hello world’’);`

  Example 1: write a “hello world” program which sends the message to the serial monitor every 500 ms. ✓

- Chance: `random(min, max)` → returns an integer between min and max-1.

  Example 2: display on serial monitor a random integer between 1 and 100, every 500 ms. Each time we call this function, we get a seemingly "random" value min ≤ value ≤ max-1.

- The need for `randomSeed(seed)`. We can supply a different *seed* value to the random number generator by measuring an unconnected analog pin (we measure noise).

  Example: random.ino

Receiving data from serial monitor

Why? We can configure our project in non-trivial ways.

- `Serial.read()`: returns -1 if no data available, otherwise returns the first byte read (byte = 8 bit integer).

  Example 3: read one byte and echo it back to the serial monitor.

```
...
void loop() {
    Serial.println(Serial.read());
    delay(1000);
}
```

- Read the byte into a char variable.

  see echo.ino

→ println will output an integer if the type of the expression to be output is int

→ output a character, if the type of the expression is char.

Example 4: write an Arduino program that receives characters from the the serial monitor and outputs the following message

probing

action.

input character	message
A .. Z	uppercase
a ... z	lowercase
none of the above	not a letter

three states for our system

starting state

send every 1 sec.

Solution in letters.ino.

Output only when character received: if statement

↳ extend echo.ino to not output □.

boolean function → true if characters are available from serial mon.
→ false if no characters are available

Example 4: `Serial.available()` - output only when a character is received.

Read a string (text value) from serial monitor

String: sequence of characters. Can be stored into a *String* variable.

- Append characters to a String variable.

  Example 5: read data as a string from serial monitor and echo it back.

- while loop syntax:

```
while ( boolean expression ) {
    • instruction 1;
    • instruction 2;
    ⋮
}
```

→ the boolean expression is evaluated. If it is true, instructions 1, 2, ... are executed. The boolean expression is evaluated again. If true, instructions 1, 2, ... are executed. E.T.C. When the boolean expression is false, the while() instruction terminates

Task: read a sequence of characters into a String variable.

Instructions:

as long as characters are available, read each character and add it at the end of the string variable.

Implement a function:

```
String readString ( ) {
    ...
}
```

Examples of while() loops:

(1)
```
while (true) {
  Serial.println("Hello");
}
```
→ infinite loop: outputs "Hello" for ever.

true : boolean expression

(2)
```
while (false) {
  Serial.println("Hello");
}
```
→ nothing is printed. The while loop terminates.

③ Write a while loop statement that outputs 1, 2, 3, ..., 10

```
x = 1;
while ( x <= 10 ) {
  Serial.println(x);
  x = x + 1;
}
```

initialization (starting point)

test (stop condition)

change (counter variable changes)

Homework: (translate)

Instructions (hint: use variables in the English instructions)

1) Assign 1 to variable x.

2) As long as x is smaller or equal to 10:
   - output x
   - increase the value of x by 1.

Task: read a sequence of characters into a String variable.

Instructions:

as long as characters are available, read each character ① and add it at the end of the string variable ②.

translation

```
char c;
String s;
while (Serial.available()) {
    c = Serial.read();      // ①
    s = s + c;              // ②
}
```

where do we put this code?

Inside a function named readString():

```
String readString() {
    [ ]
    return s;
}
```

← s is the value of the function

Count from 1 to 10:

```
x = 1;
while ( x <= 10 ) {
  Serial.println(x);
  x = x+1;
}
```

initialization (starting point)

test (stop condition)

change (counter variable changes)

for loops →

```
for ( int x = 1 ; x <= 10 ; x = x+1 )
{
  Serial.println(x);
}
```

Syntax for loops:

```
for ( initialization ; test ; change ) {
  - instr 1
  :
}
```

order of evaluation is exactly as for the equivalent while loop.

Read a numeric value from serial monitor

- `str.toInt()` (*str* is a String variable): Read a string, then convert it to an int.

  Example 6: Define a *readInt()* function. Use it to seed the random number generator instead of using the value of noise.

  read int
  ↳ read a string
  ↳ convert it to int.

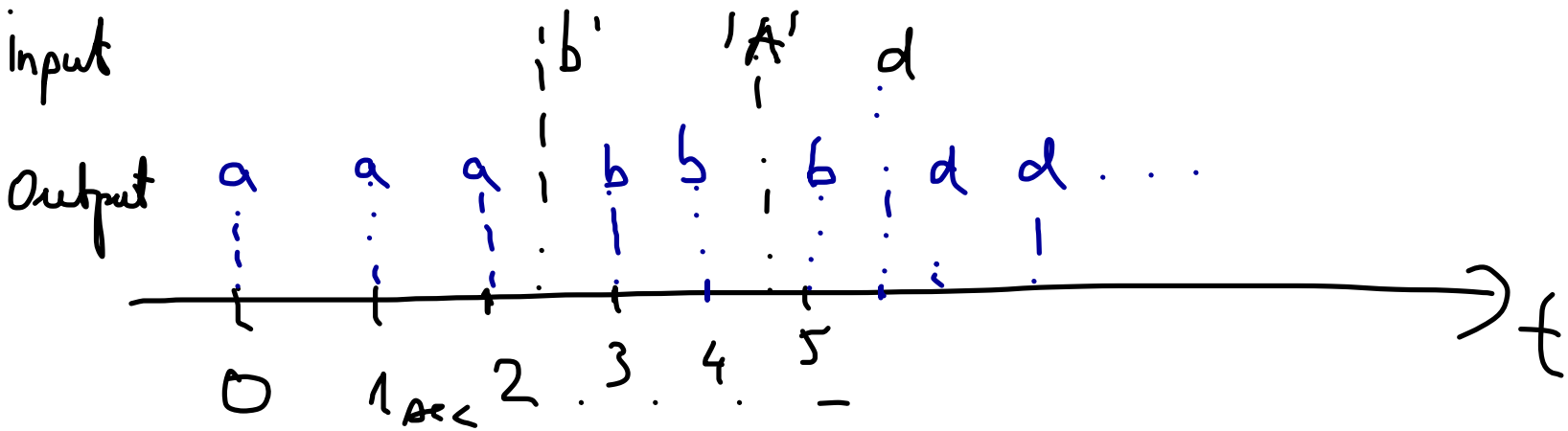
  To convert to float instead, look for the correct function in reference.

Exercises

- Blink a LED, but make the LED skip every fourth blink.
- Make the LED be on and off for a period of time that changes randomly.
- Connect 10 LEDs to digital pins 2-11. Light one LED at a time, simulating a "left to right" motion. Hint: use a for loop to control the on/off pattern.

Homework assignments

1) Arduino outputs 'a' every 1 sec. It also accepts input.
If the input char. is a ... z then it will output that character.
If the input character is not a .. z (e.g., Z, B, 1, ';' ...) it will not change its behaviour.



Template

State or global variables
→ the character to output
(declare this variable as char)

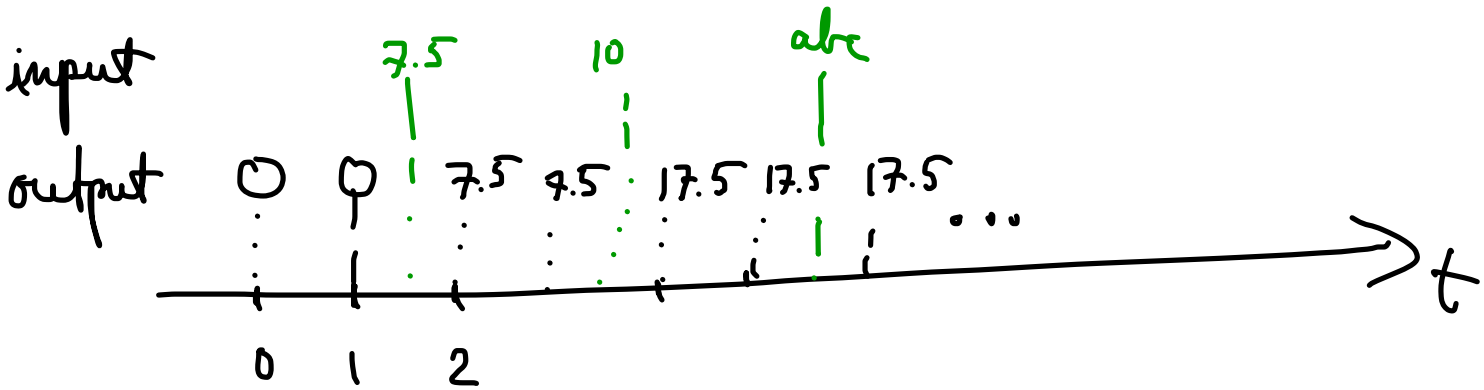
```
void loop() {
```

- read a character if it is available (not in state variable yet) ← probing
- if char. read is 'a'..'z' then change the state variable to that char.

- output the state variable ← action
- wait 1 sec.

```
}
```

Problem 2 : Write a program that adds floating point values and displays the sum every sec.



Template

State of system:
- the sum: global, initially 0

```
void loop() {
```

- read floating point value from Serial monitor.
- adding the value to the sum

← probing

- output the sum
- wait 1 sec.

← action

```
}
```

How do we read a floating point value from the serial monitor?

Re-use readString & modify readInt to create a function called readFloat().

readString.ino . . .