WorkShop

Integration of scientific and engineering practices into STEM approach using Arduino

Psycharis Sarantos, Professor
ASPETE

Papazoglou Panayotis, Associate Professor
Technological Education Institute of Sterea Ellada (Central Greece)

Main Goal of the workshop

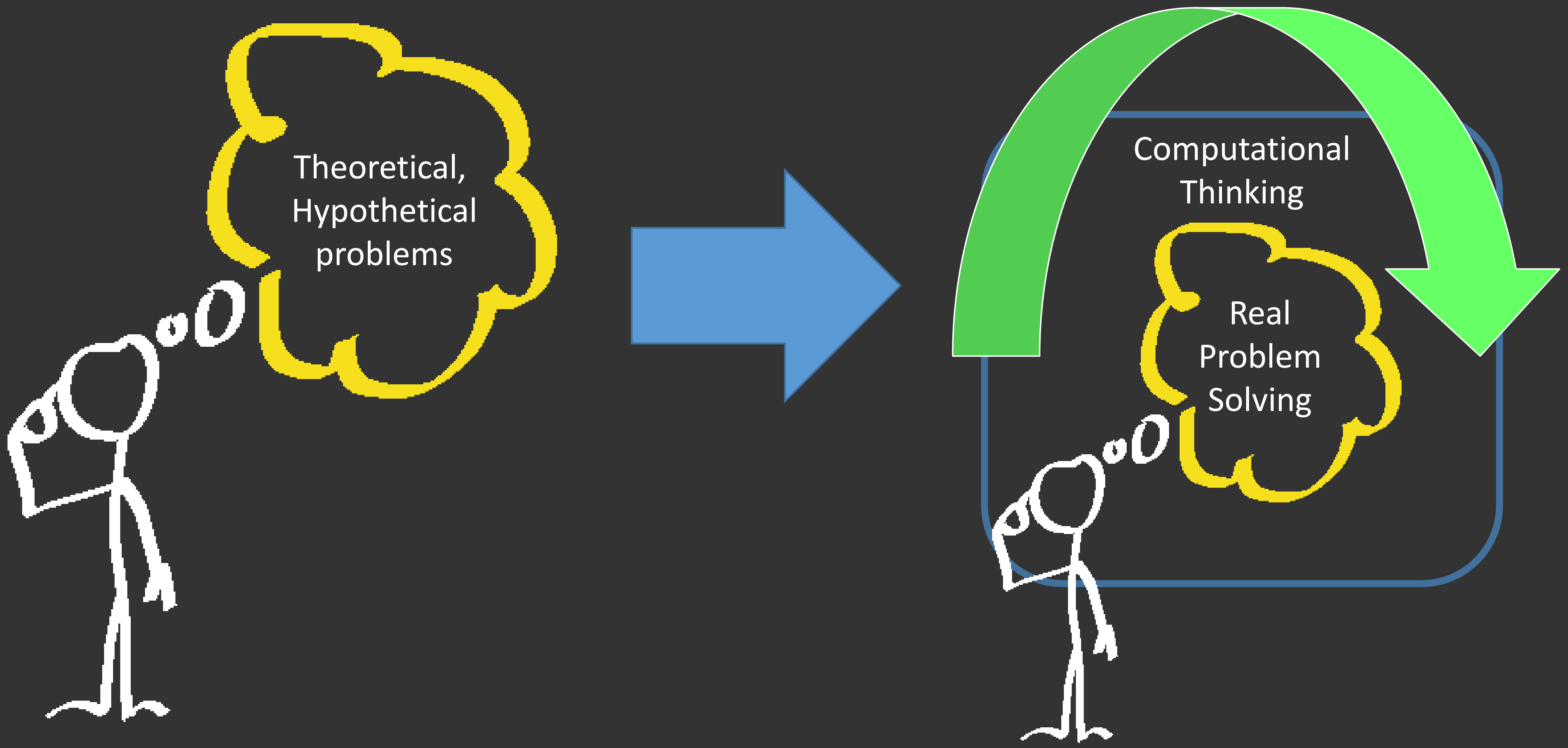
To work on specific practical applications which can be approached from different educational points of view based on adapted skills and knowledge of elementary, middle and high school students through the STEM-CT disciplines

WorkShop outline

- STEM & CT : A new Approach in school
- Application 1: Real Time Temperature Monitoring (RTTM)
  - Level 1: RTTM using the S4A software (Elementary school)
  - Level 2: RTTM using mean value in serial plotter (Elementary/Middle school)
- Application 2: Electronic dice
  - Level 1: “Rolling” the dice with a button (Middle/High school)
  - Level 2: “Distance” dice “Rolling” using an ultrasonic sensor (Middle/High school)

The applications will be implemented in different levels based on student needs, skills and knowledge

STEM



Theoretical, Hypothetical problems

Computational Thinking

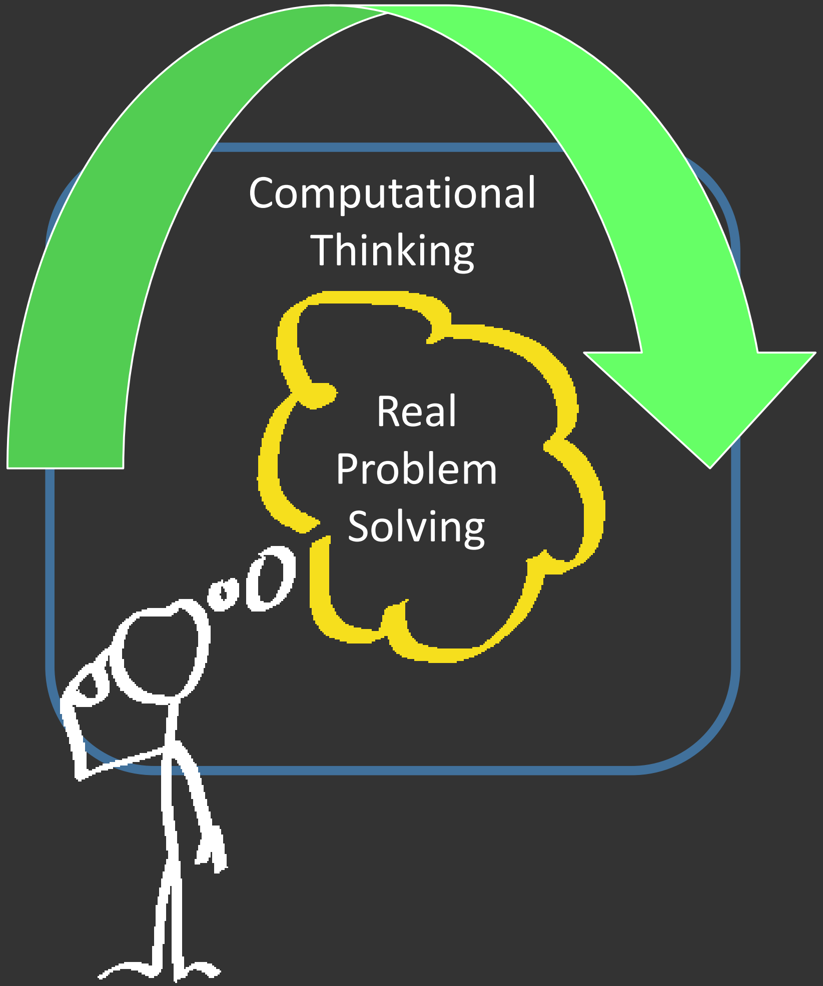
Real Problem Solving

Traditional school

A new era in school

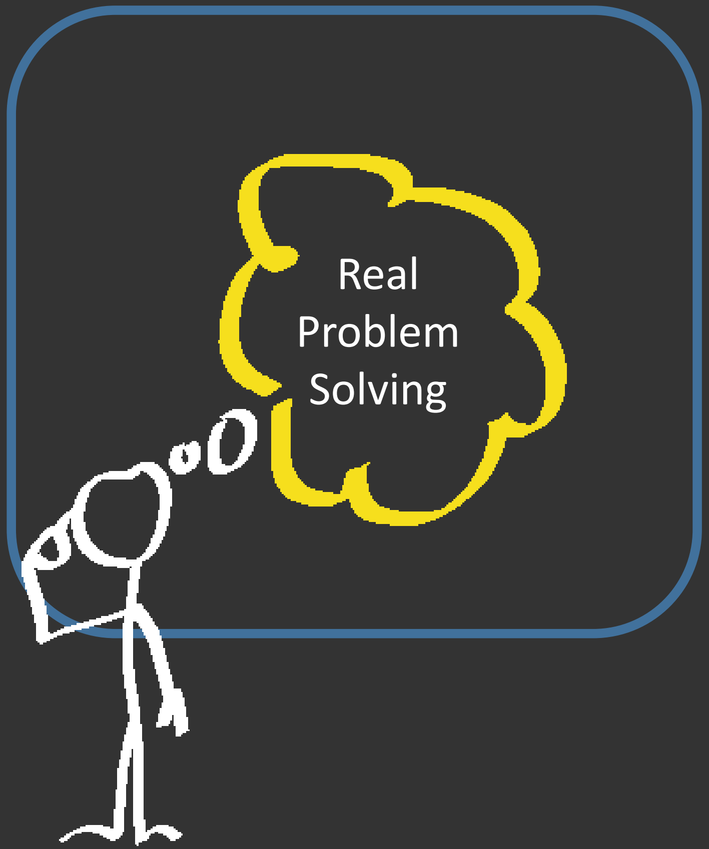
(a new way of thinking)

STEM



- Skill emerging
- Constructive and creative thinking
- Real life problems
- Knowledge component synthesis
- Fulfilling future world requirements

CT-Computational Thinking



- A new way of thinking for problem solving and gaining new knowledge

- Can be “applied” to any type of problems

- Some core concepts:
    - Algorithm
    - Abstraction
    - Decomposition
    - Pattern recognition

Core questions/goals for STEM in education

- Which Skills have to be emerged ?

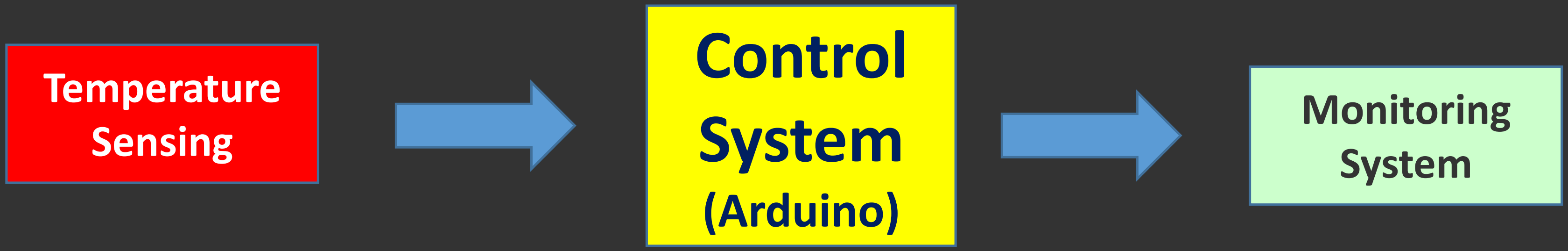
- How the new knowledge is adapted to existing knowledge ?

- How the STEM disciplines can be adapted to student needs based on the corresponding class level ?

- Each STEM application has to be approached from different educational point of view based on the real student needs

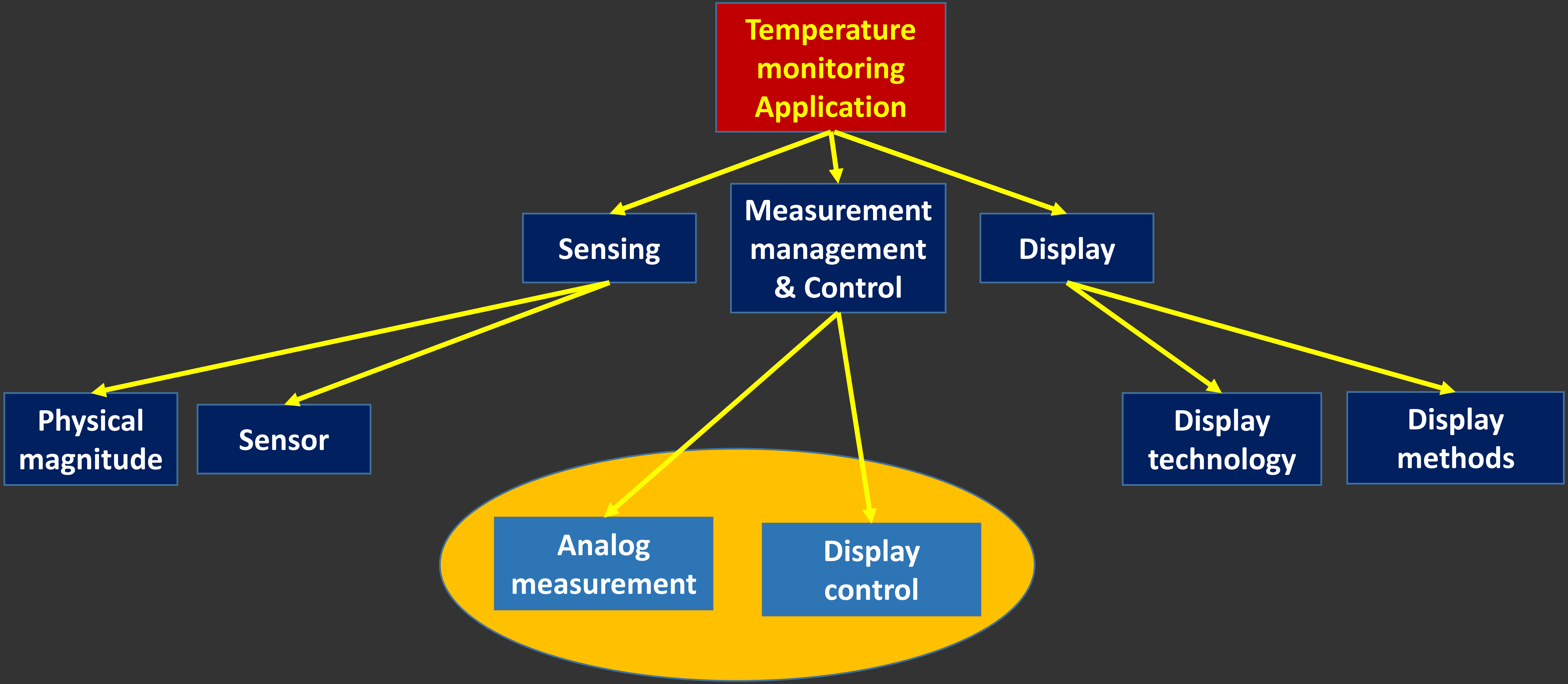
Application1: Real time temperature monitoring

Application design at abstract level



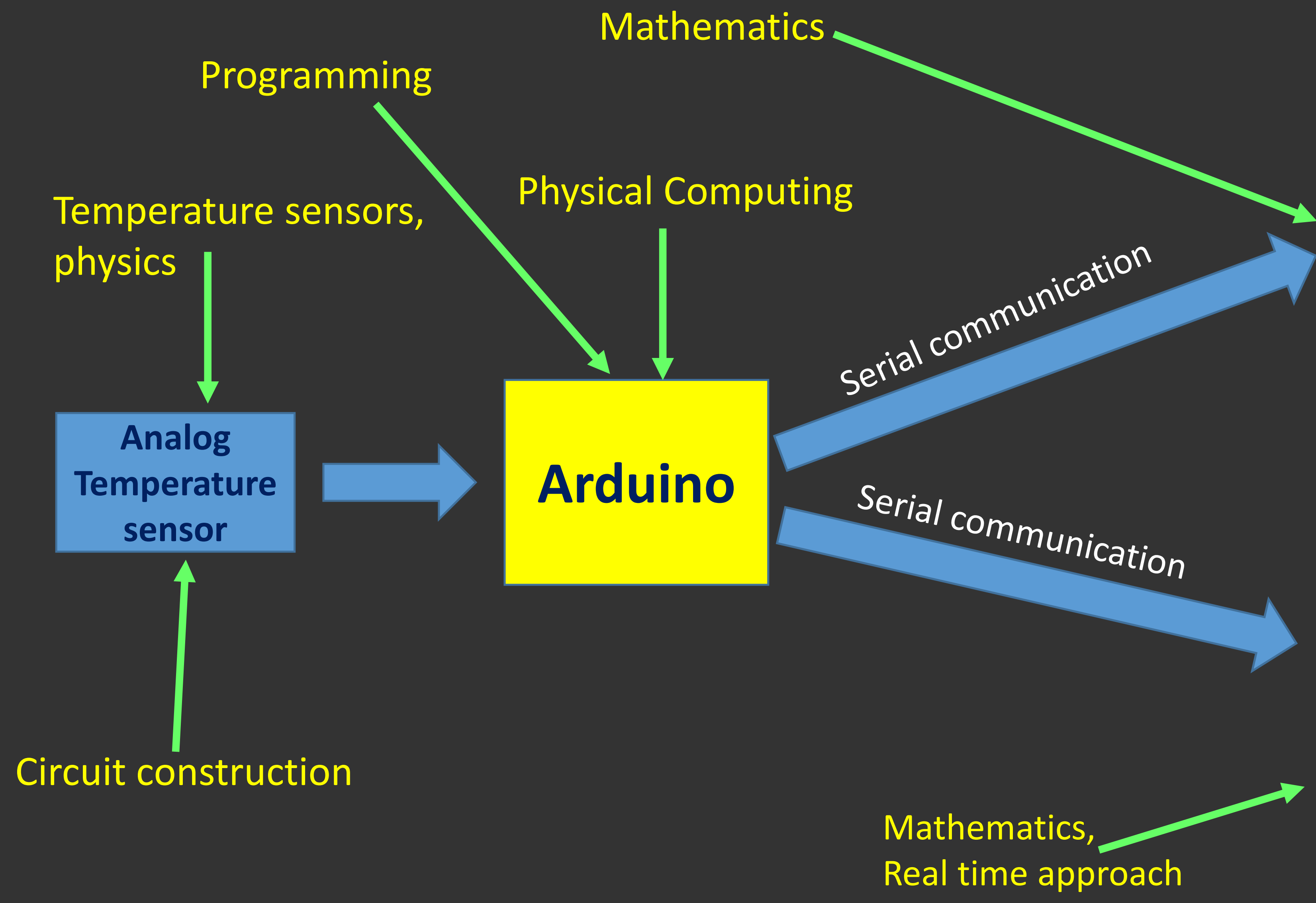
Application: Real time temperature monitoring

Decomposition level – Application components



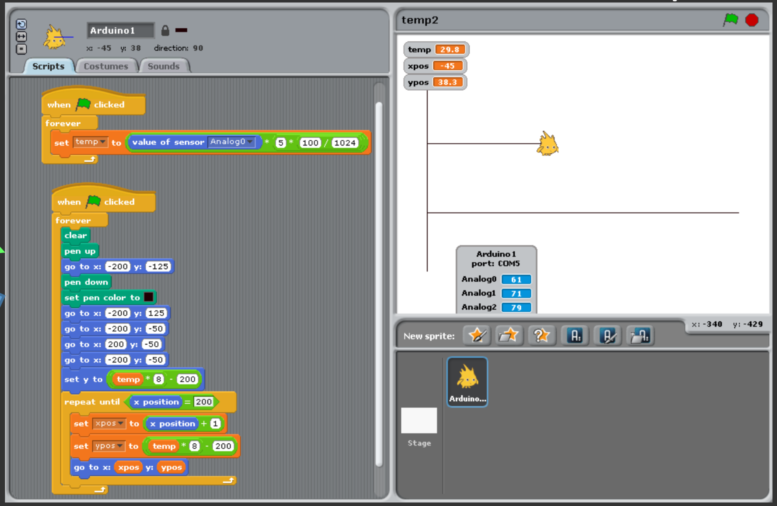
Application/Problem core components

Which Skills & Knowledge are suitable for your class?



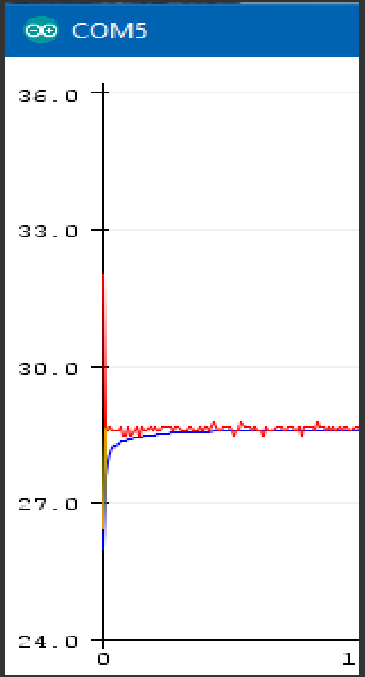
Programming

Computer



```
when clicked
forever
  set temp to value of sensor Analog0 * 5 * 100 / 1024

when clicked
forever
  clear
  pen up
  go to x: -200 y: -125
  pen down
  set pen color to
  go to x: -200 y: 125
  go to x: -200 y: -50
  go to x: 200 y: -50
  go to x: -200 y: -50
  set y to temp * 8 - 200
  repeat until x position = 200
    set xpos to x position + 1
    set ypos to temp * 8 - 200
    go to x: xpos y: ypos
```

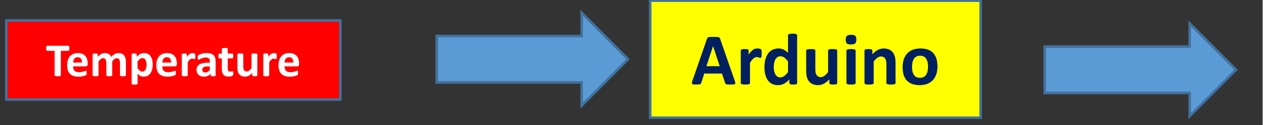
Computer

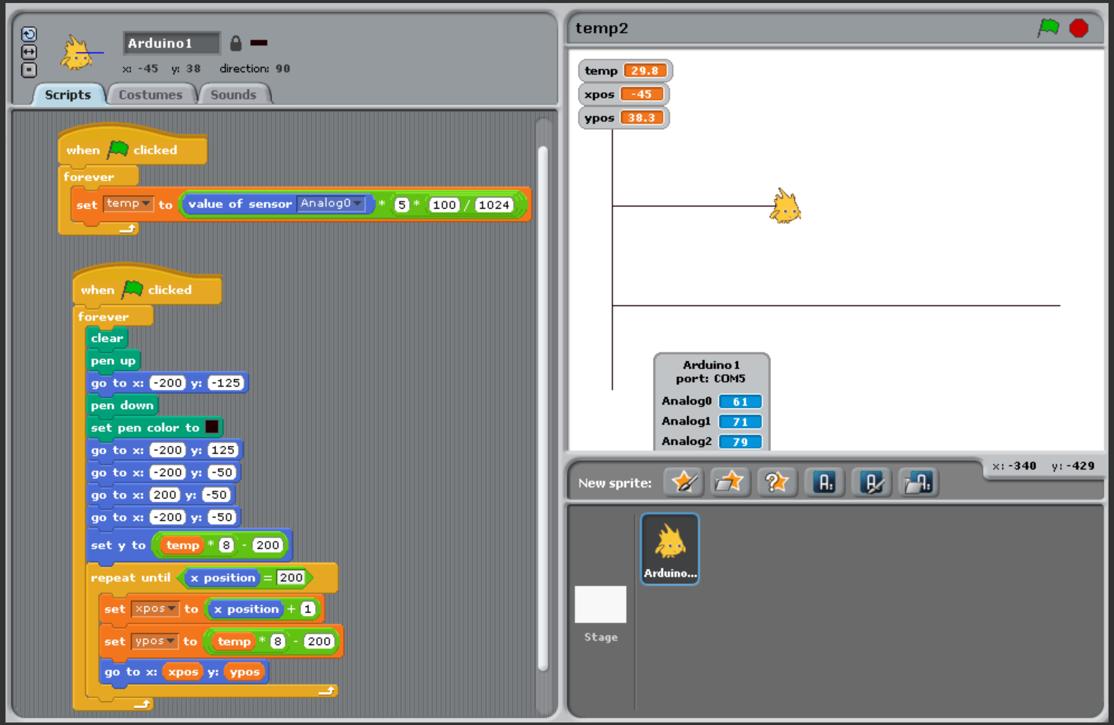


Serial plotter
(Arduino IDE)

Elementary School

Level 1: Real time monitoring with S4A





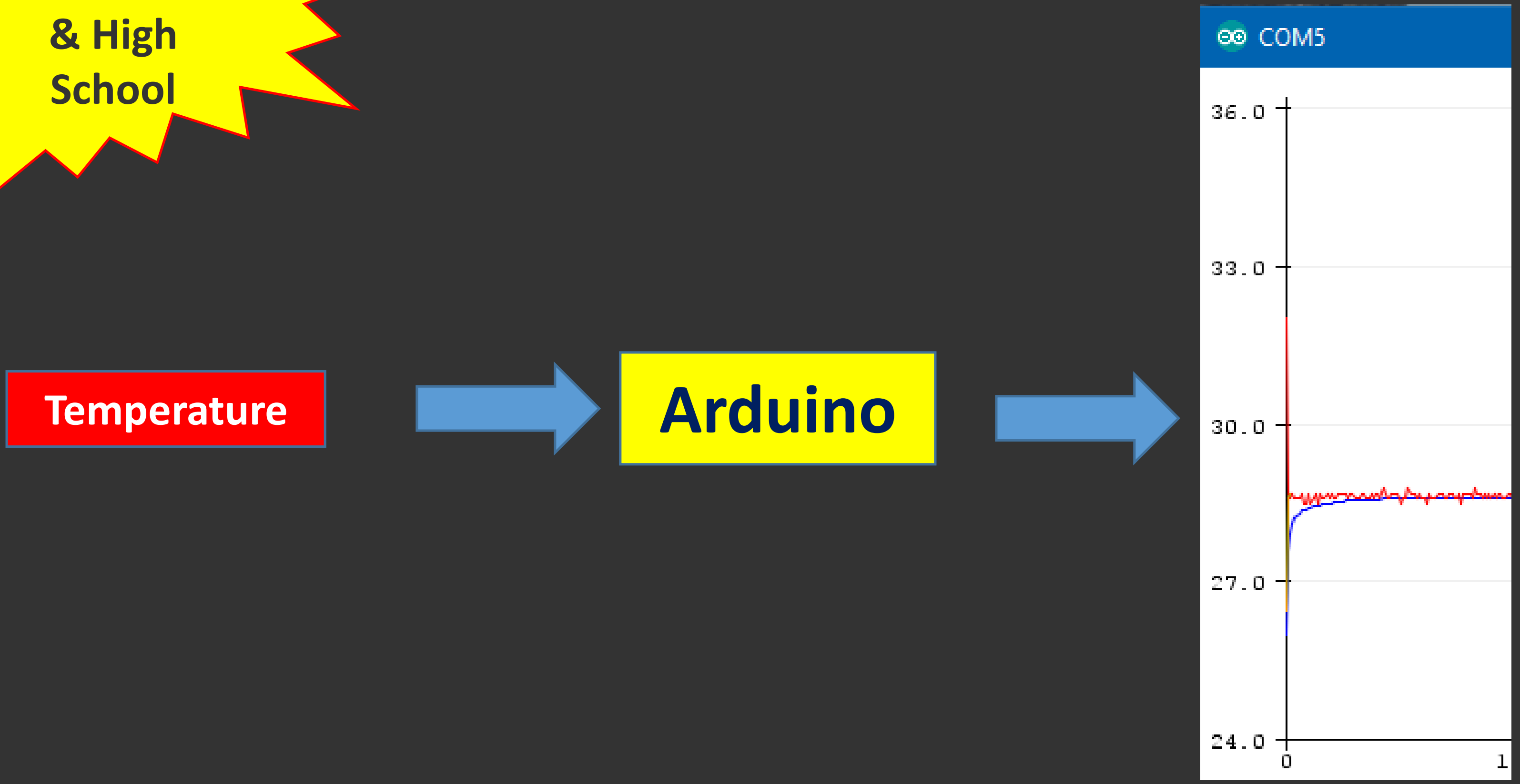
```
when ⚑ clicked
forever
    set temp to value of sensor Analog0 * 5 * 100 / 1024

when ⚑ clicked
forever
    clear
    pen up
    go to x: -200 y: -125
    pen down
    set pen color to ■
    go to x: -200 y: 125
    go to x: -200 y: -50
    go to x: 200 y: -50
    go to x: -200 y: -50
    set y to temp * 8 - 200
    repeat until x position = 200
        set xpos to x position + 1
        set ypos to temp * 8 - 200
        go to x: xpos y: ypos
```

```
temp 29.8
xpos -45
ypos 38.3

Arduino1
port: COM5
Analog0 61
Analog1 71
Analog2 79
```

Skills/Knowledge: Algorithm/Program development, mathematics, circuits, physical computing

Level 2: Real time monitoring using mean value

Middle
& High
School



Temperature

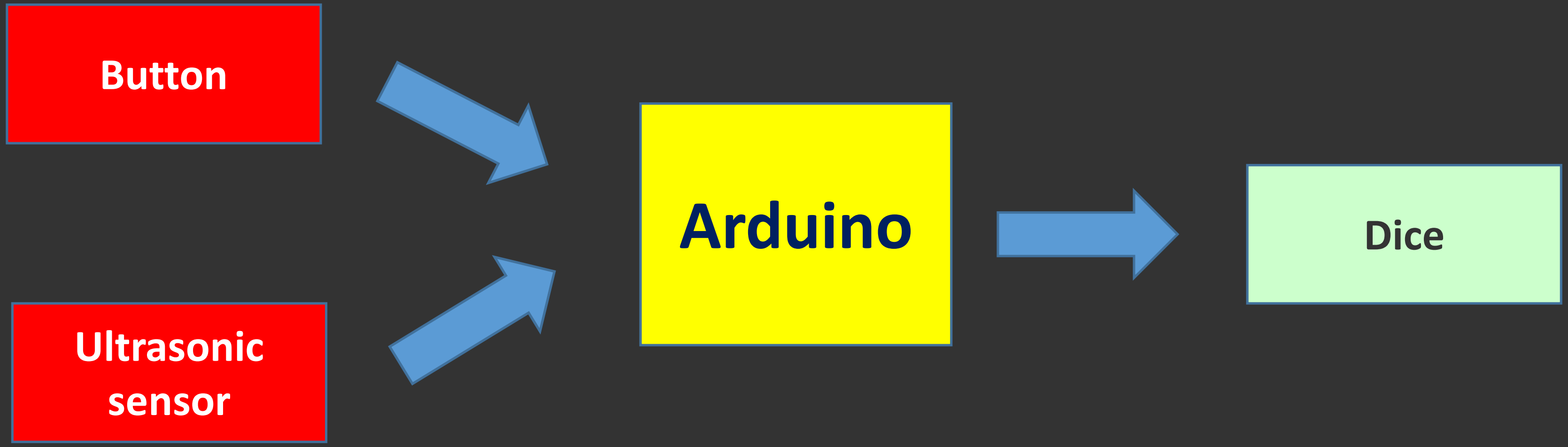
Arduino

🟥 Current temperature

🟦 Current mean

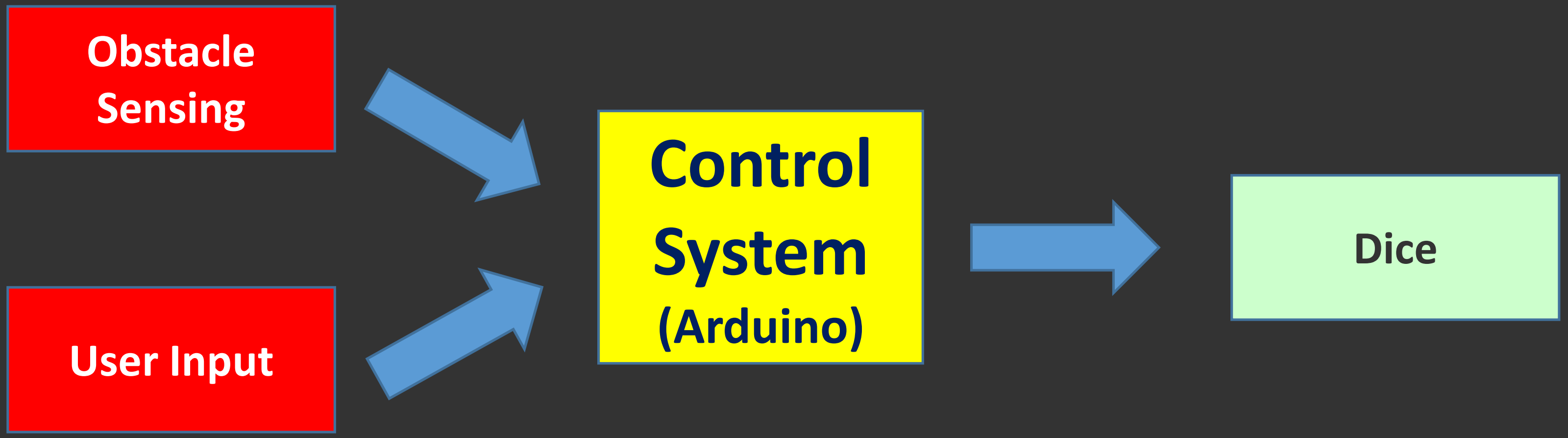
Skills/Knowledge: Algorithm/Program development, mathematics, circuits, physical computing

Application 2: Electronic dice



Application2: Electronic dice

Application design at abstract level



Obstacle Sensing

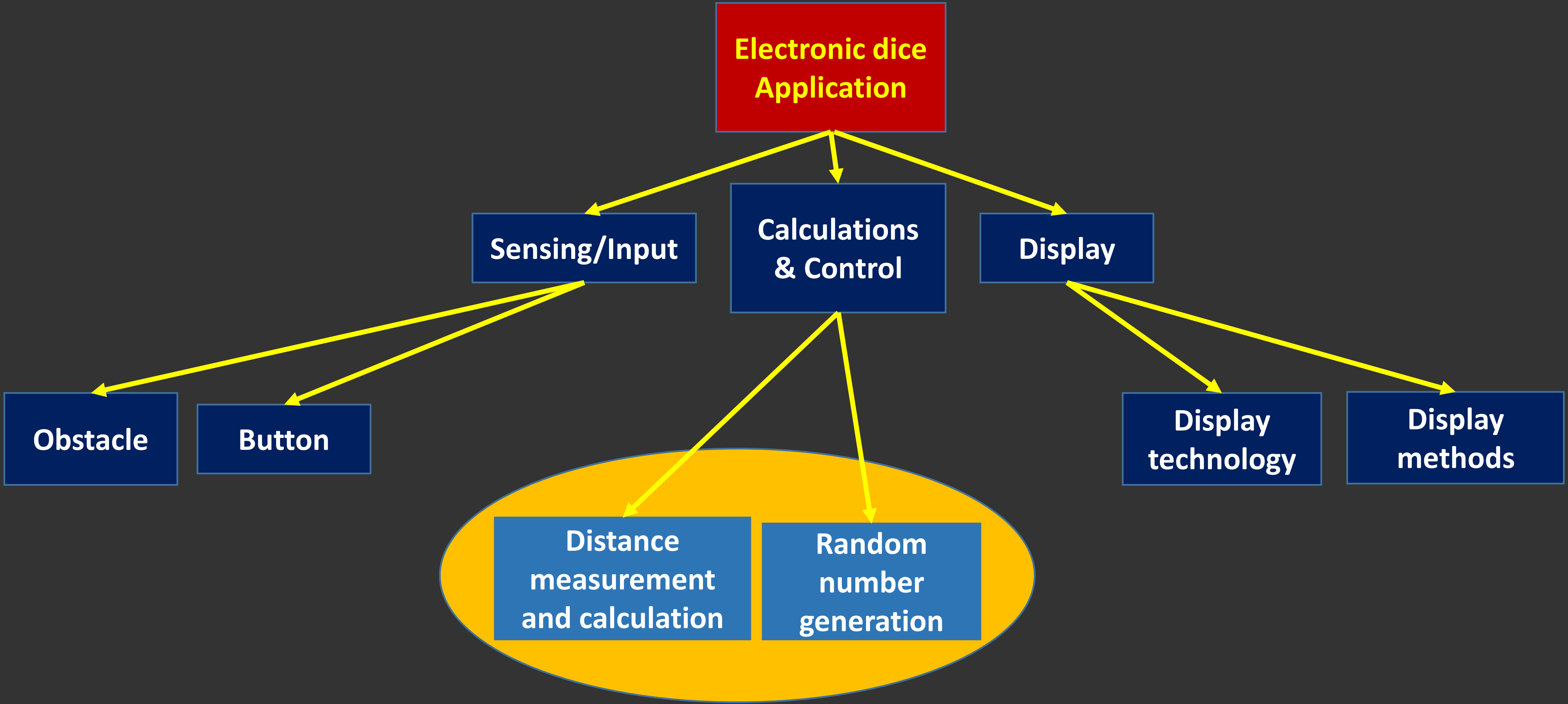
User Input

Control System (Arduino)

Dice

Application: Electronic dice

Decomposition level – Application components



Electronic dice Application

Sensing/Input

Calculations & Control

Display

Obstacle

Button

Distance measurement and calculation

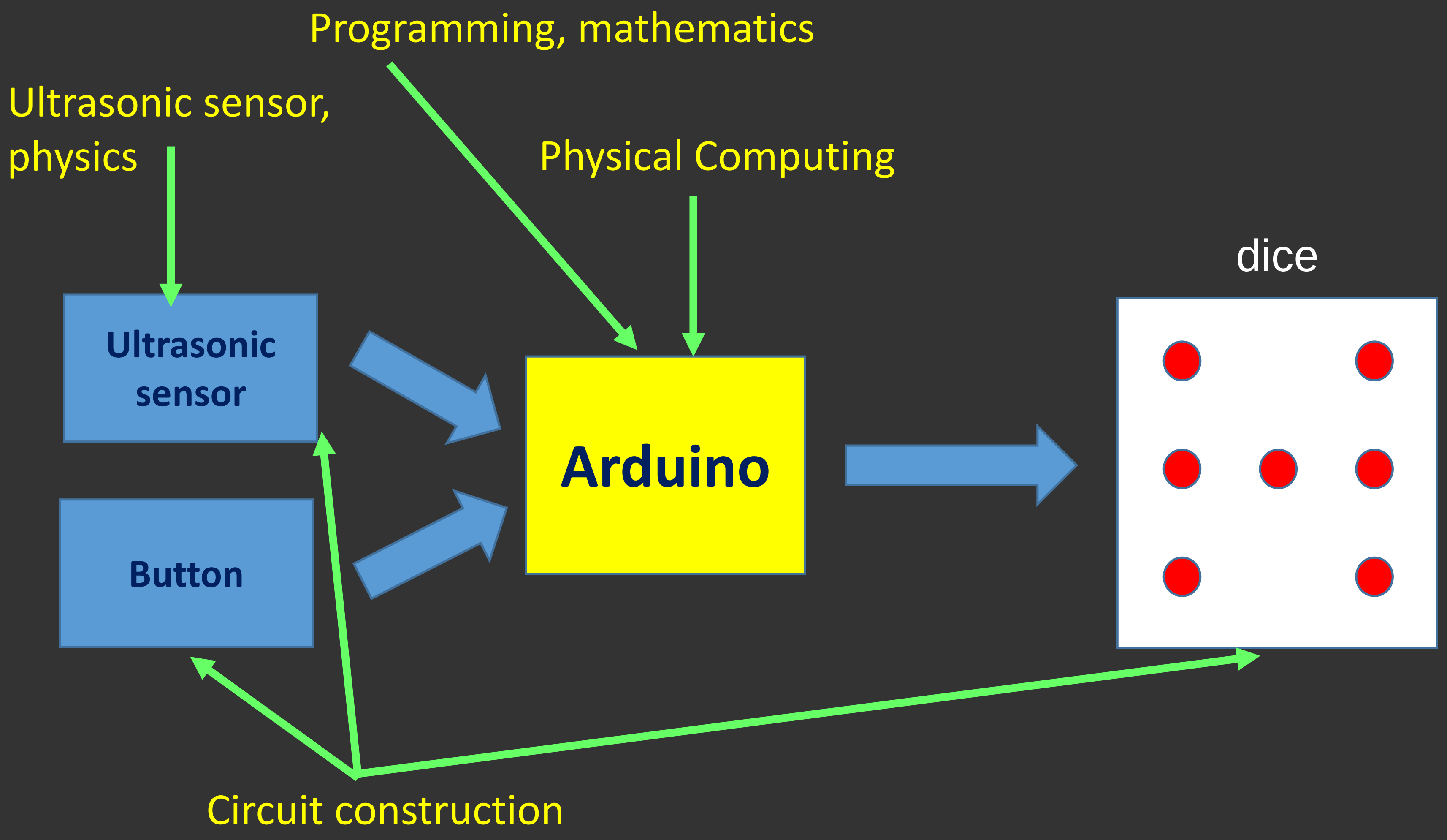
Random number generation

Display technology

Display methods

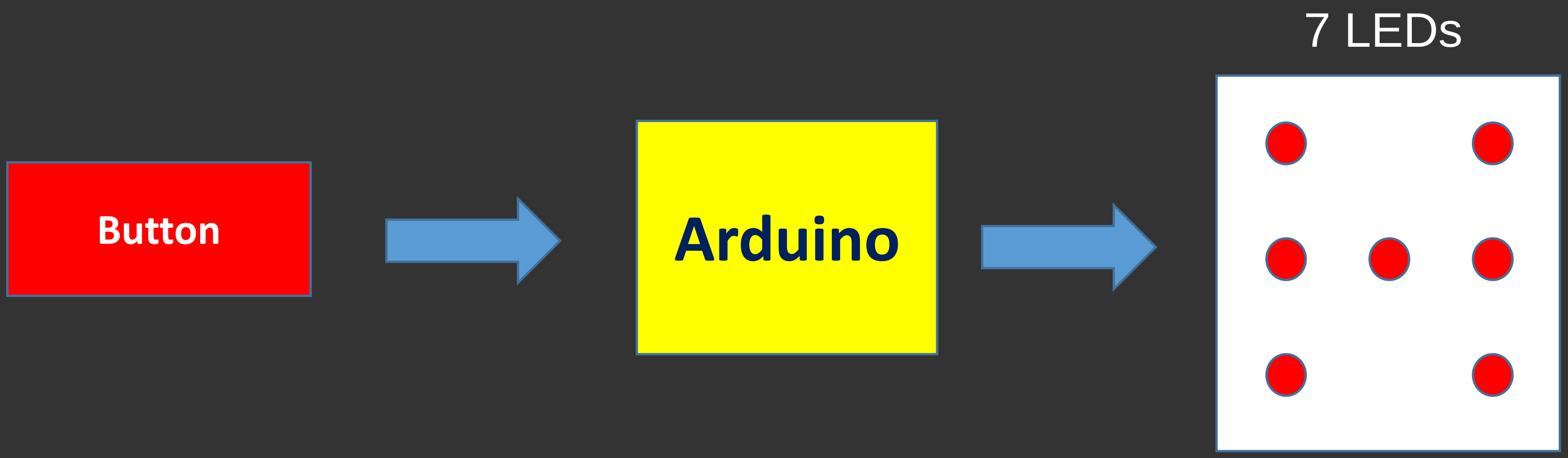
Application/Problem core components

Which Skills & Knowledge are suitable for your class?



Level 1: “Rolling” the dice with a button

Middle
& High
School



Button

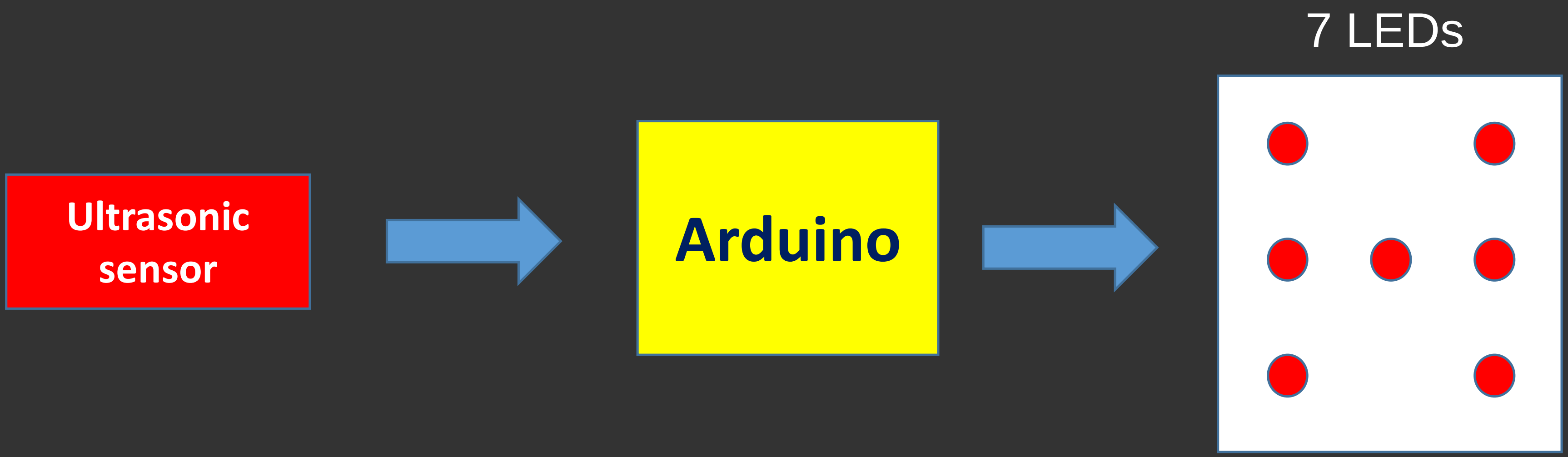
Arduino

7 LEDs

Skills/Knowledge: Algorithm/Program development, circuits, physical computing

Level 2: “Distance” dice “Rolling” using an ultrasonic sensor

Middle
& High
School



Ultrasonic sensor

Arduino

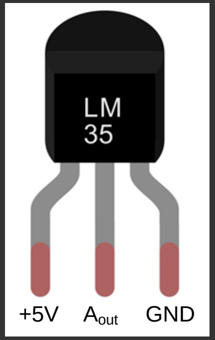
7 LEDs

Skills/Knowledge: Algorithm/Program development, physics, circuits, physical computing

Implementation: Real time monitoring

Level 1: Real time monitoring with S4A (1)

Temperature



1

Analog signal

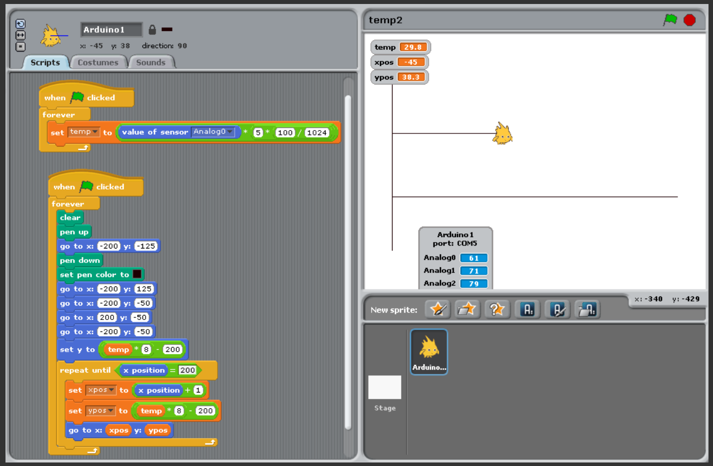
Arduino

S4A Firmware

2

Serial com

Computer (S4A)



```
when clicked
forever
    set temp to value of sensor Analog0 * 5 * 100 / 1024

when clicked
forever
    clear
    pen up
    go to x: -200 y: -125
    pen down
    set pen color to
    go to x: -200 y: 125
    go to x: -200 y: -50
    go to x: 200 y: -50
    go to x: -200 y: -50
    set y to temp * 8 - 200
    repeat until x position = 200
        set xpos to x position + 1
        set ypos to temp * 8 - 200
        go to x: xpos y: ypos
```

temp 29.8
xpos -45
ypos 38.3

Arduino1
port: COM5
Analog0 61
Analog1 71
Analog2 79

3

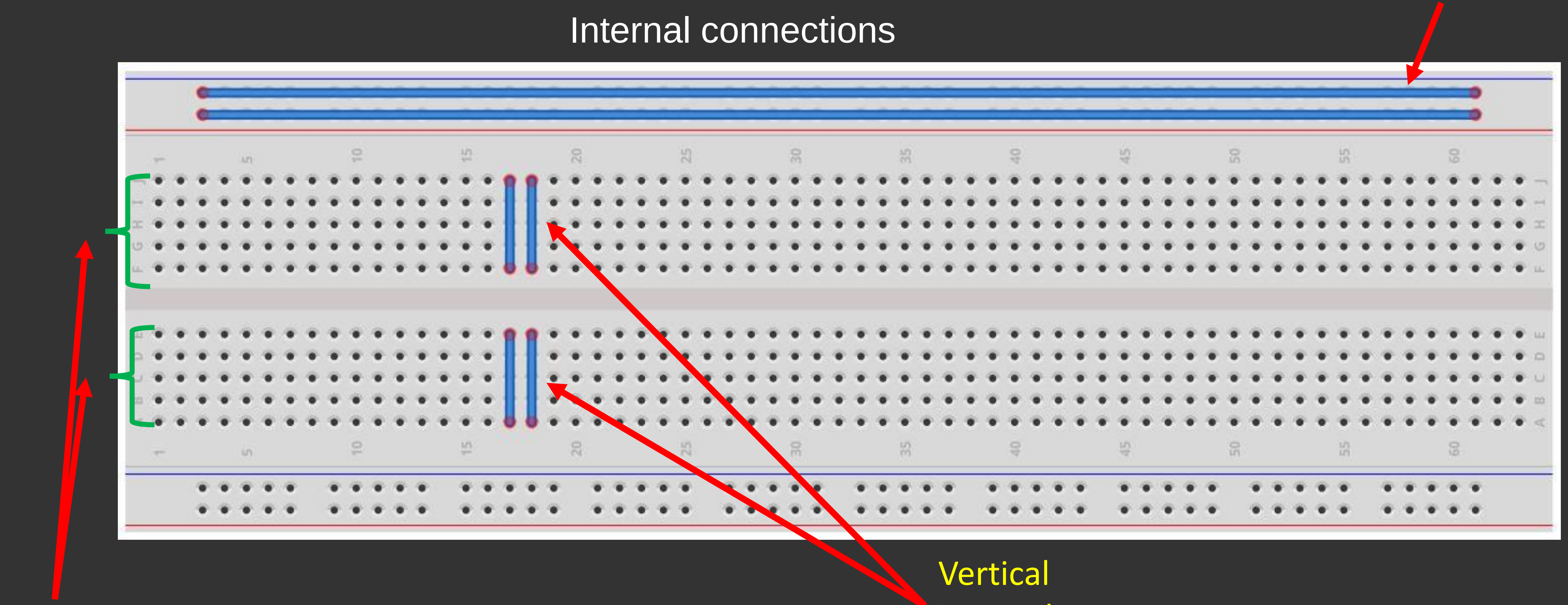
1: LM35, Temperature Analog Sensor, Output: 10mV/^{o}C

2: Upload S4A Firmware in Arduino

3: Program development in S4A

Using the breadboard

Internal connections



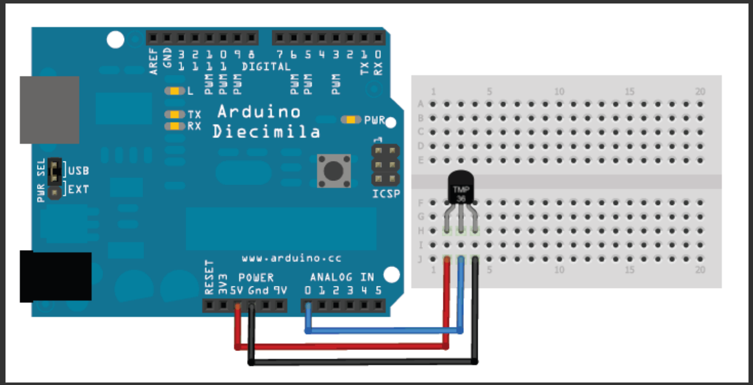
Horizontal internal connections

Vertical internal connections

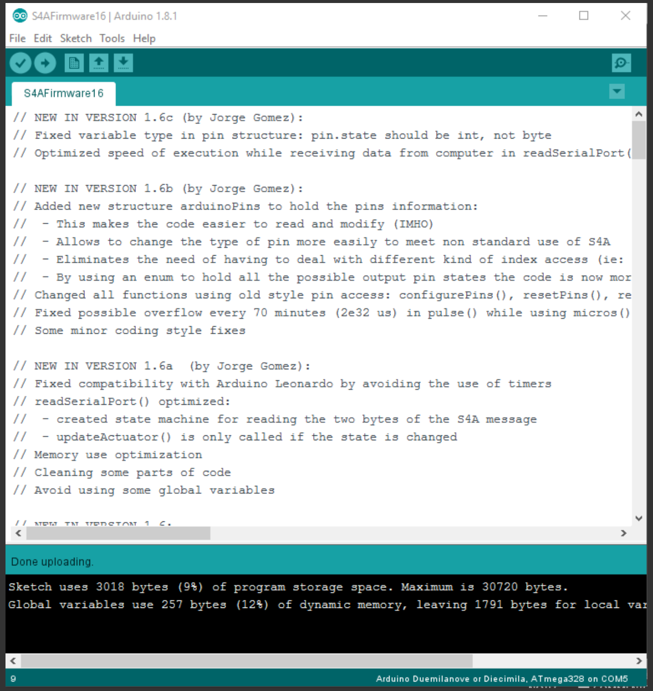
Separated Areas

Level 1: Real time monitoring with S4A (2)

STEP 1 Connect the circuit (LM35)



STEP 2 Upload S4A Firmware



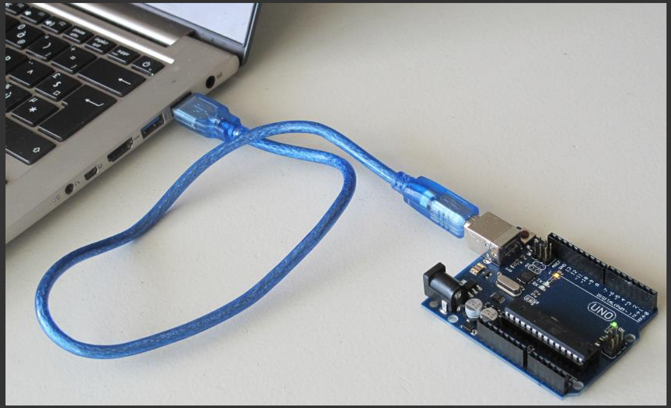
```
// NEW IN VERSION 1.6c (by Jorge Gomez):
// Fixed variable type in pin structure: pin.state should be int, not byte
// Optimized speed of execution while receiving data from computer in readSerialPort(

// NEW IN VERSION 1.6b (by Jorge Gomez):
// Added new structure arduinoPins to hold the pins information:
//  - This makes the code easier to read and modify (IMHO)
//  - Allows to change the type of pin more easily to meet non standard use of S4A
//  - Eliminates the need of having to deal with different kind of index access (ie:
//  - By using an enum to hold all the possible output pin states the code is now mor
// Changed all functions using old style pin access: configurePins(), resetPins(), re
// Fixed possible overflow every 70 minutes (2e32 us) in pulse() while using micros()
// Some minor coding style fixes

// NEW IN VERSION 1.6a  (by Jorge Gomez):
// Fixed compatibility with Arduino Leonardo by avoiding the use of timers
// readSerialPort() optimized:
//  - created state machine for reading the two bytes of the S4A message
//  - updateActuator() is only called if the state is changed
// Memory use optimization
// Cleaning some parts of code
// Avoid using some global variables
```

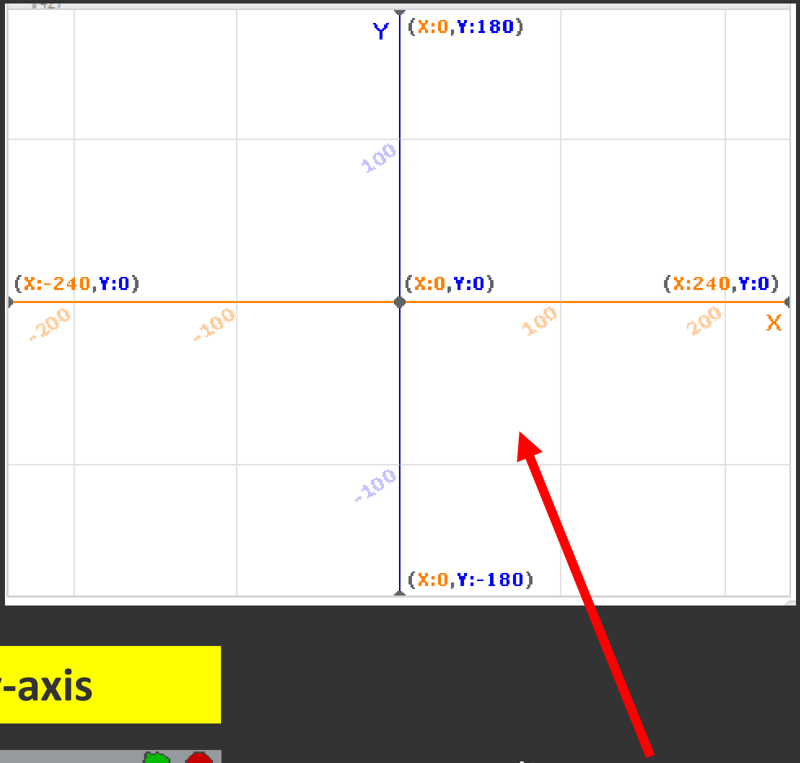
Done uploading.

```
Sketch uses 3018 bytes (9%) of program storage space. Maximum is 30720 bytes.
Global variables use 257 bytes (12%) of dynamic memory, leaving 1791 bytes for local var
```

9 Arduino Duemilanove or Diecimila, ATmega328 on COM5

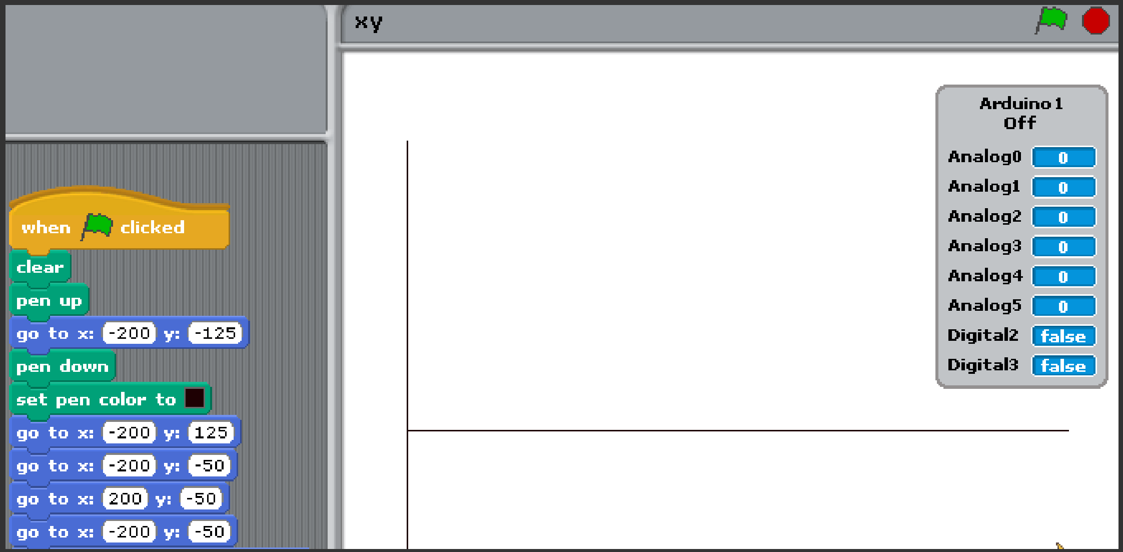
STEP 3 Connect Arduino to PC





S4A Coordinate system

STEP 4A Sketching x-axis and y-axis



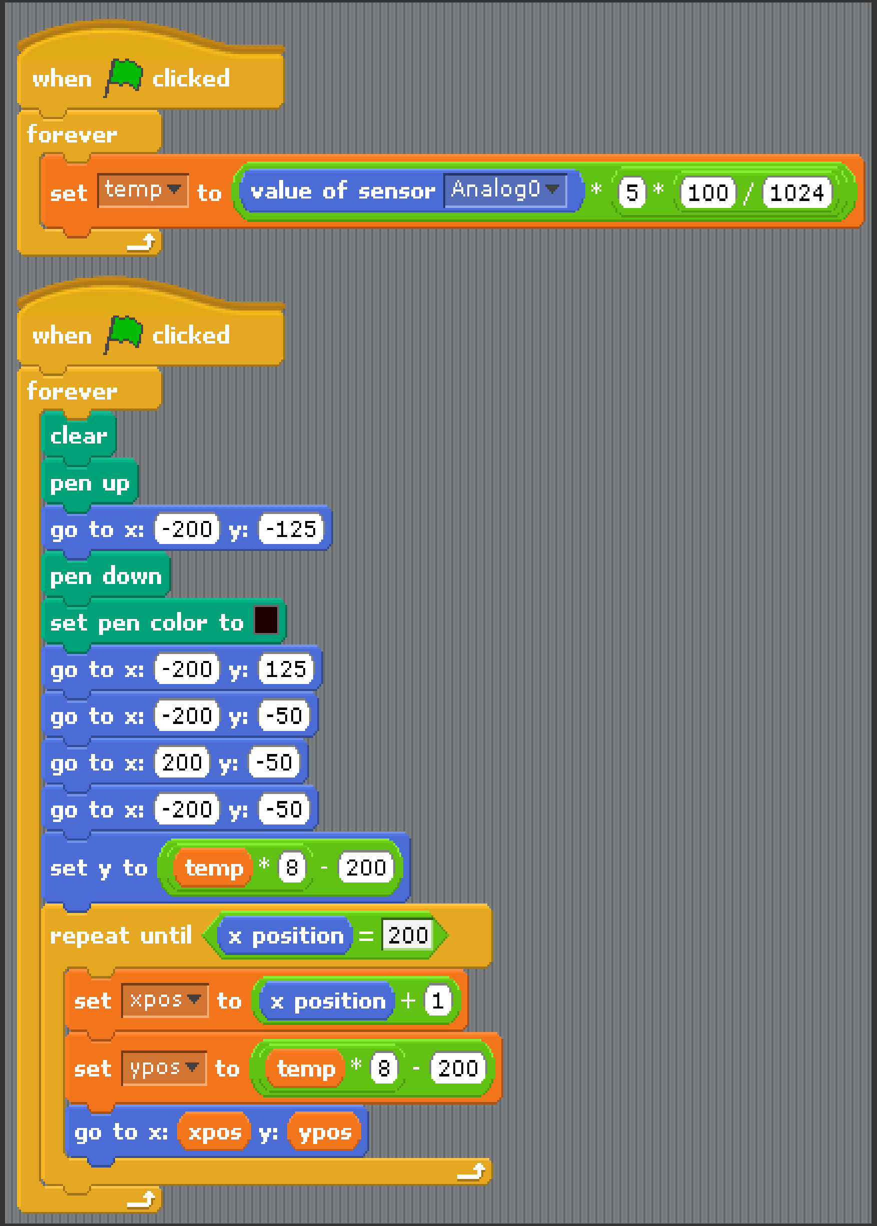
```
when 🏴 clicked
clear
pen up
go to x: -200 y: -125
pen down
set pen color to ■
go to x: -200 y: 125
go to x: -200 y: -50
go to x: 200 y: -50
go to x: -200 y: -50
```

Arduino1
Off

Analog0	0
Analog1	0
Analog2	0
Analog3	0
Analog4	0
Analog5	0
Digital2	false
Digital3	false

Level 1: Real time monitoring with S4A (3)

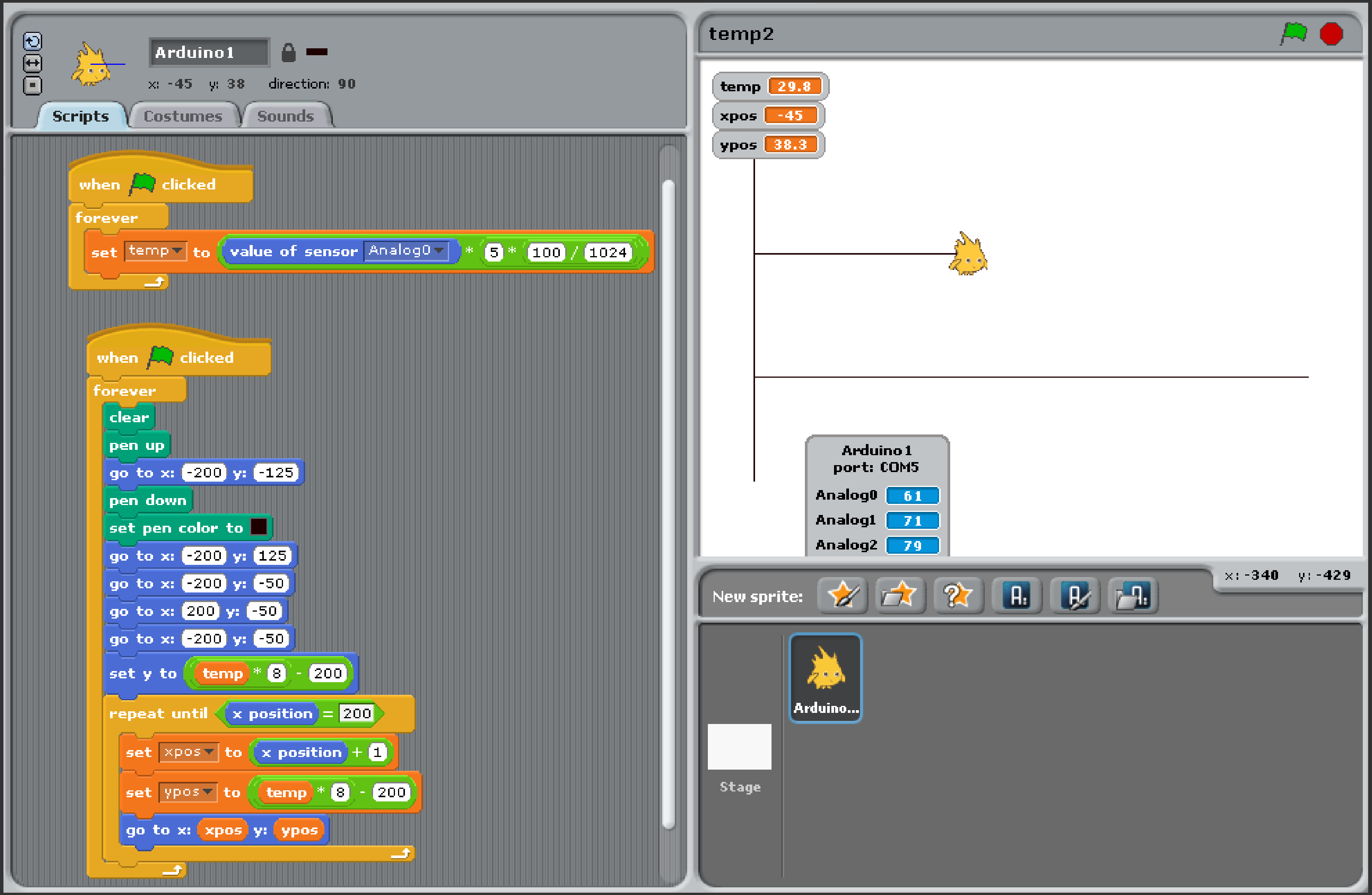
STEP 4B Develop full algorithm



```
when ⚑ clicked
forever
    set temp ▾ to value of sensor Analog0 ▾ * 5 * 100 / 1024

when ⚑ clicked
forever
    clear
    pen up
    go to x: -200 y: -125
    pen down
    set pen color to ■
    go to x: -200 y: 125
    go to x: -200 y: -50
    go to x: 200 y: -50
    go to x: -200 y: -50
    set y to temp * 8 - 200
    repeat until x position = 200
        set xpos ▾ to x position + 1
        set ypos ▾ to temp * 8 - 200
        go to x: xpos y: ypos
```

Level 1: Real time monitoring with S4A (4)

STEP 5 Run the program



Arduino1

x: -45 y: 38 direction: 90

```
when [flag] clicked
forever
    set temp to ( value of sensor Analog0 * 5 * 100 / 1024 )

when [flag] clicked
forever
    clear
    pen up
    go to x: -200 y: -125
    pen down
    set pen color to ■
    go to x: -200 y: 125
    go to x: -200 y: -50
    go to x: 200 y: -50
    go to x: -200 y: -50
    set y to ( temp * 8 - 200 )
    repeat until < x position = 200 >
        set xpos to ( x position + 1 )
        set ypos to ( temp * 8 - 200 )
        go to x: xpos y: ypos
```

temp2

```
temp 29.8
xpos -45
ypos 38.3
```

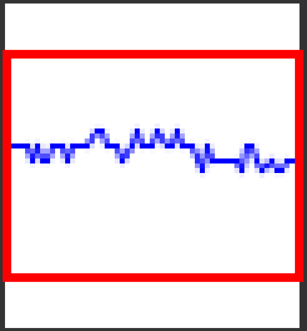
```
Arduino1
port: COM5
Analog0 61
Analog1 71
Analog2 79
```

x: -340 y: -429

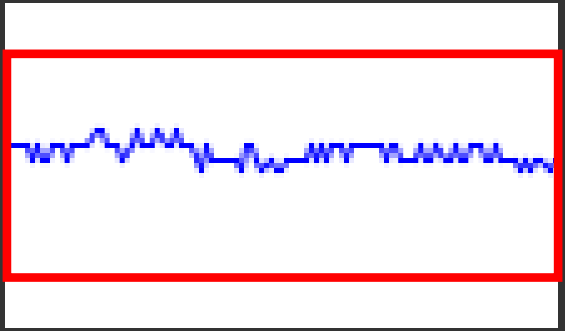
Level 2: Real time monitoring using mean value (1)

$$\overline{cm} = \frac{1}{cn}\sum_{i=1}^{cn} temp_i$$

cm = current mean value
cn = current number of samples
$temp_i$ = current temperature



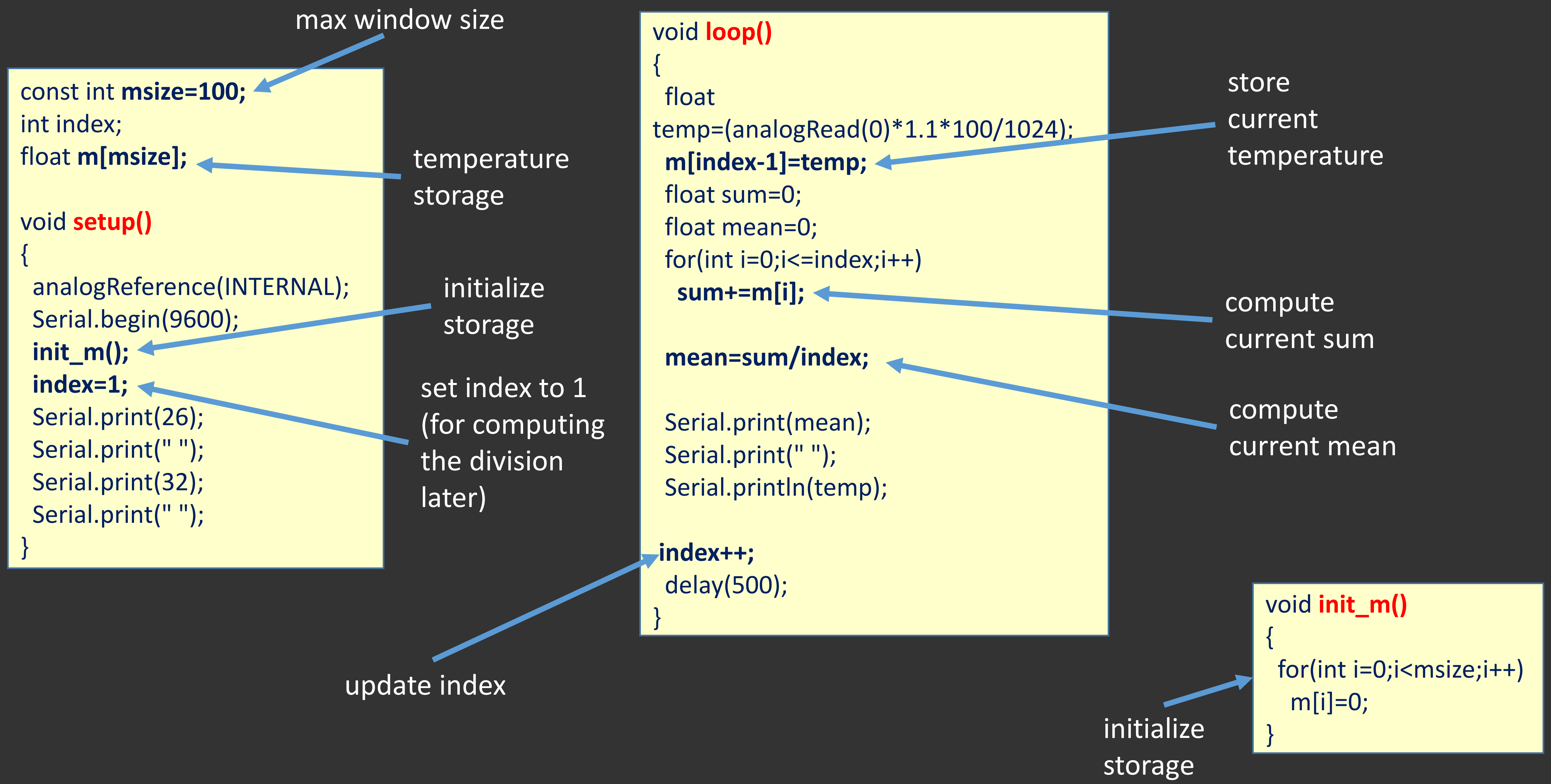
Memory window with 50 samples, cn=50, compute new cm



Memory window with 100 samples, cn=100, compute new cm

Dynamic window range
[sample 1, current sample]

Level 2: Real time monitoring using mean value (2)



```
const int msize=100;
int index;
float m[msize];

void setup()
{
 analogReference(INTERNAL);
 Serial.begin(9600);
 init_m();
 index=1;
 Serial.print(26);
 Serial.print(" ");
 Serial.print(32);
 Serial.print(" ");
}
```

```
void loop()
{
 float
temp=(analogRead(0)*1.1*100/1024);
 m[index-1]=temp;
 float sum=0;
 float mean=0;
 for(int i=0;i<=index;i++)
  sum+=m[i];

 mean=sum/index;

 Serial.print(mean);
 Serial.print(" ");
 Serial.println(temp);

index++;
 delay(500);
}
```

```
void init_m()
{
 for(int i=0;i<msize;i++)
  m[i]=0;
}
```

max window size

temperature storage

initialize storage

set index to 1 (for computing the division later)

update index

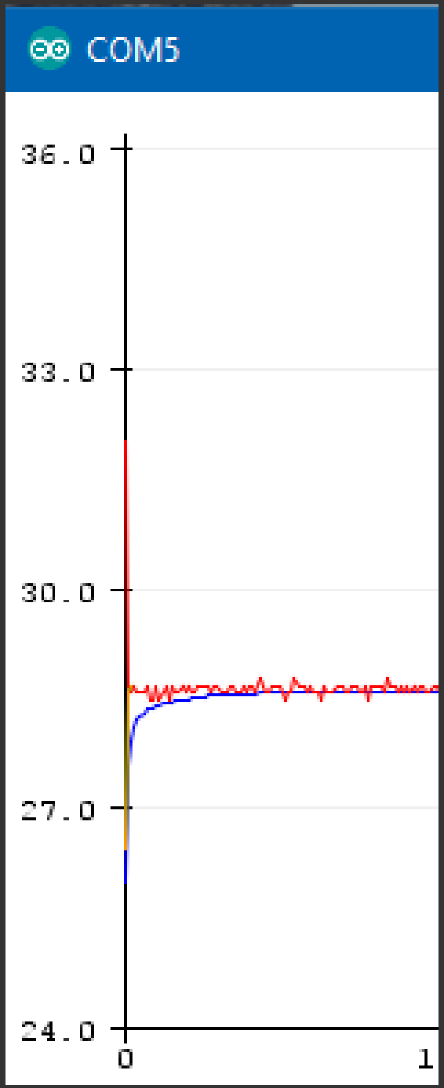
store current temperature

compute current sum

compute current mean

initialize storage

Level 2: Real time monitoring using mean value (3)



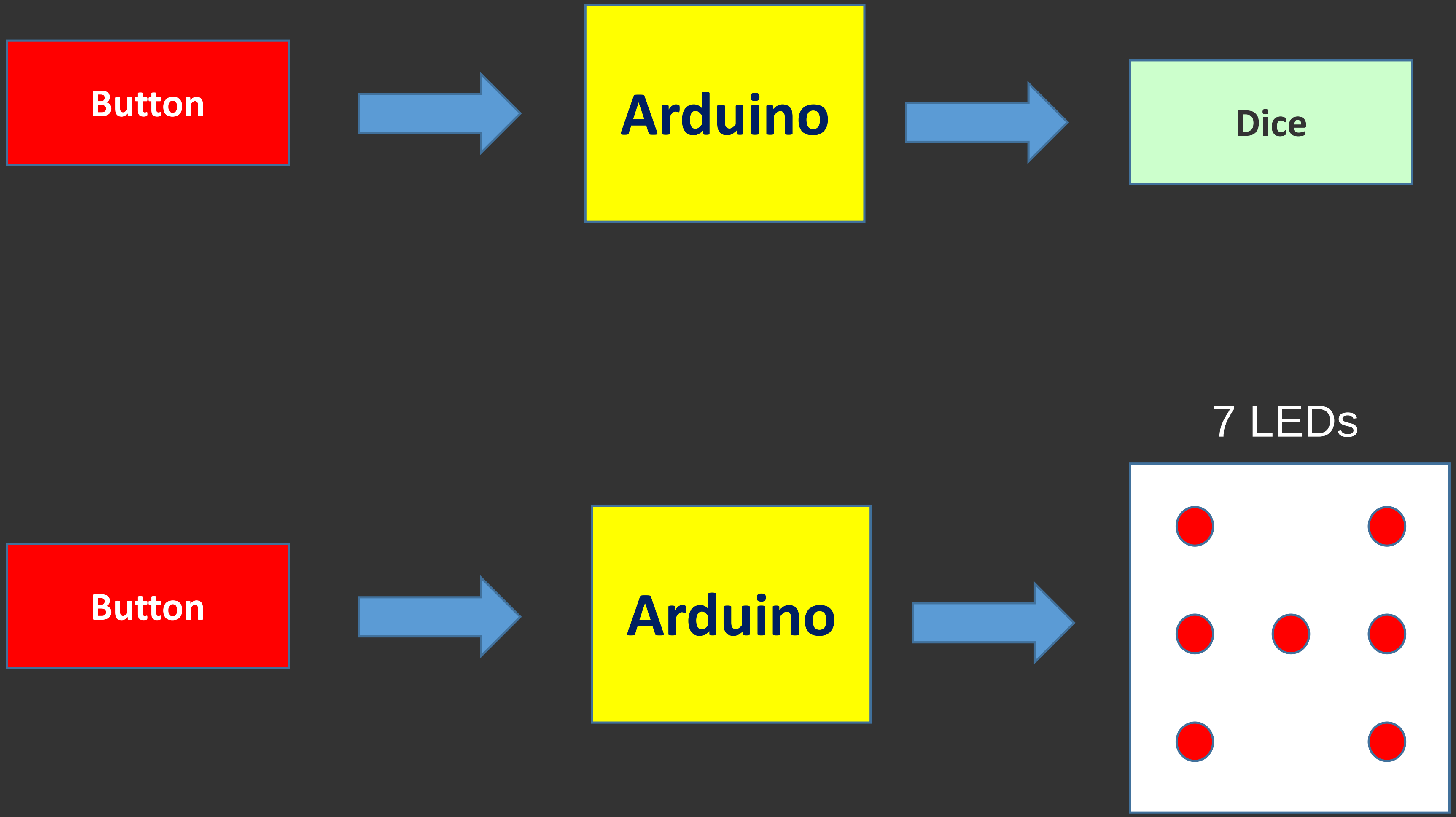
🟥 Current temperature

🟦 Current mean

Try the code
(Arduino)

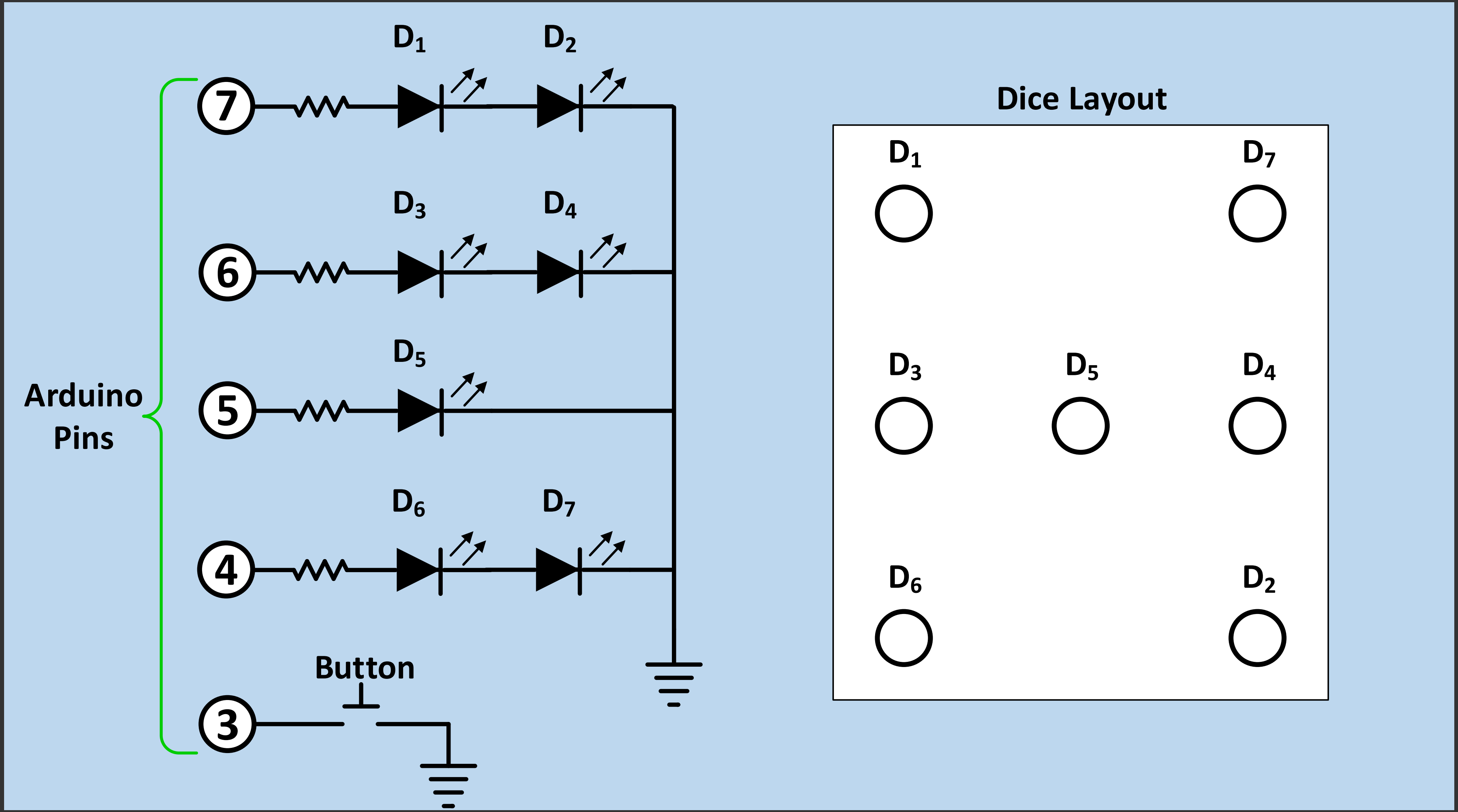
Implementation: Electronic dice

Level 1: “Rolling” the dice with a button (1)



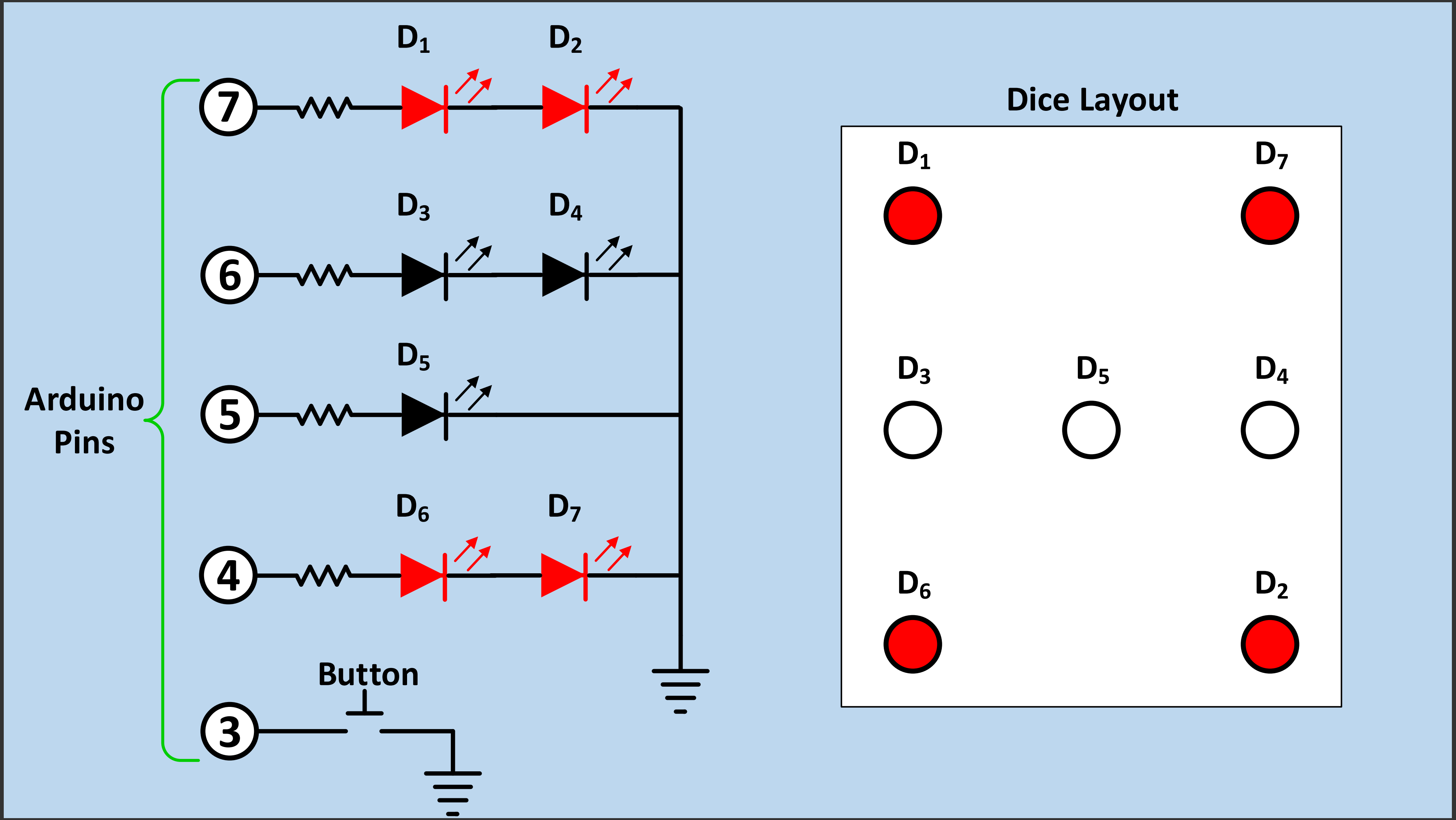
If the button is pressed, “roll” the dice…

Level 1: “Rolling” the dice with a button (1)



Electrical Circuit and LED (dice) layout

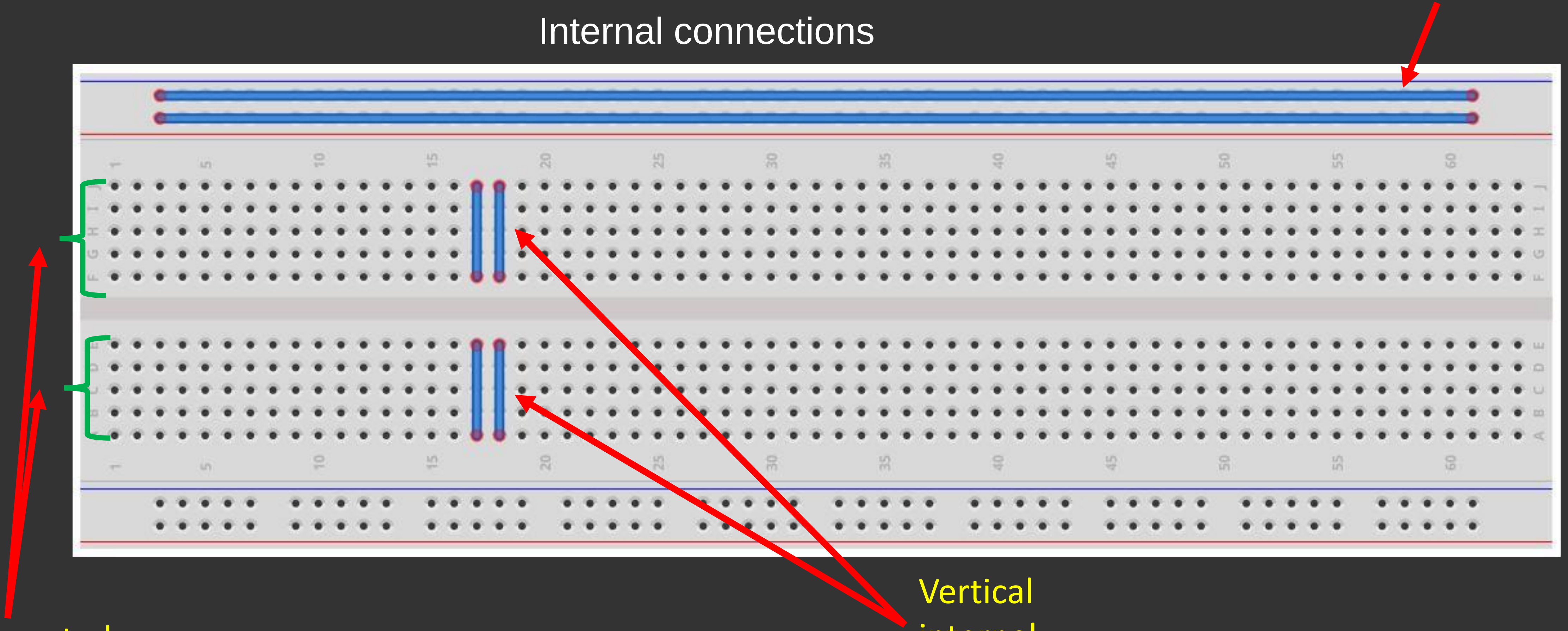
Level 1: “Rolling” the dice with a button (2)



Example: number four on the dice (active LEDs D_1, D_2, D_6 , D_7)

Using the breadboard

Internal connections

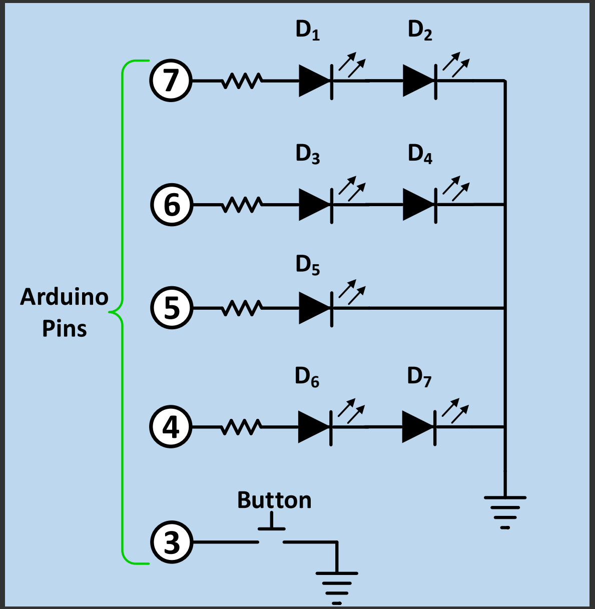


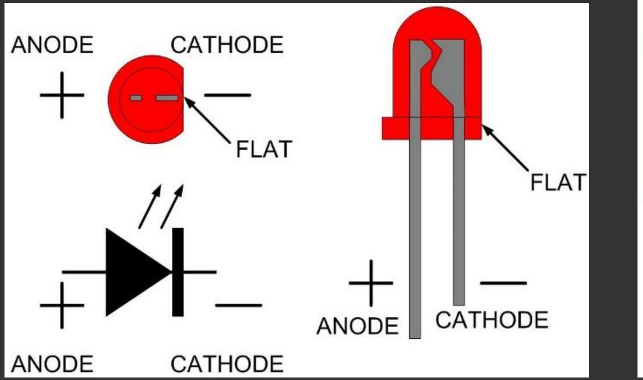
Horizontal internal connections

Vertical internal connections

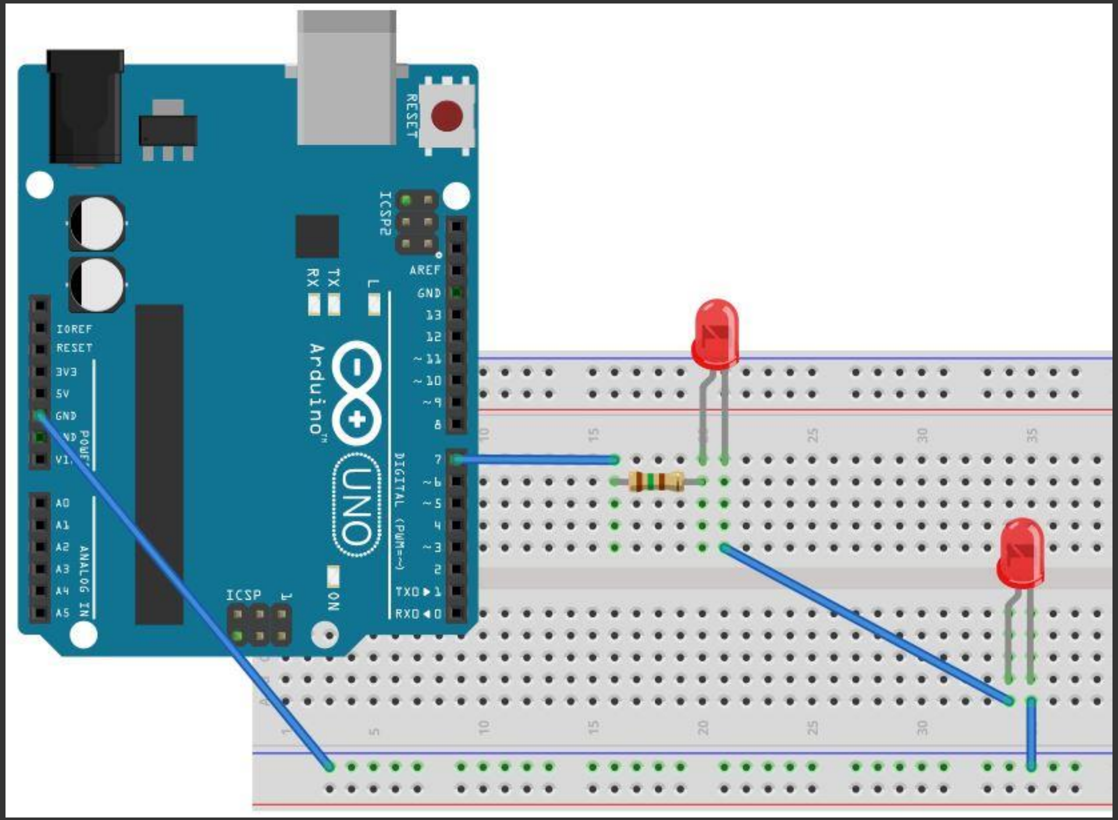
Separated Areas

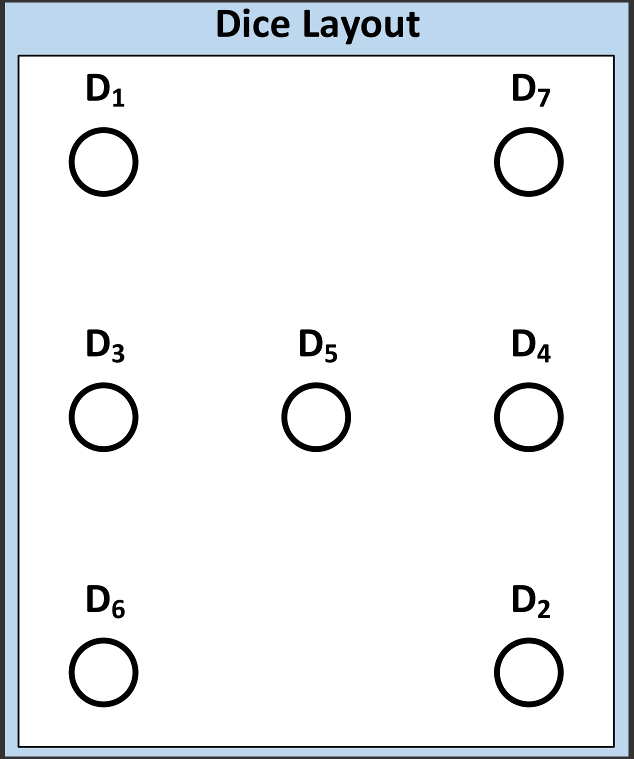
Level 1: “Rolling” the dice with a button (3)



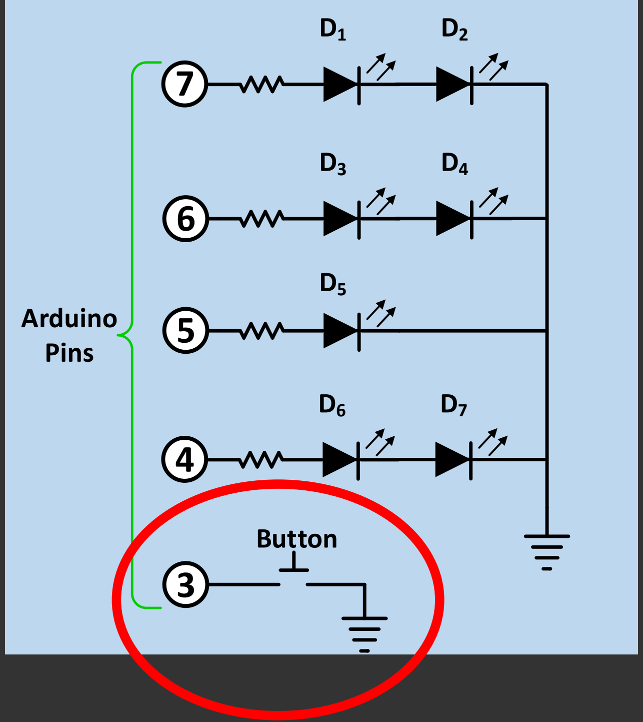


Example: connecting D$_1$, D$_2$

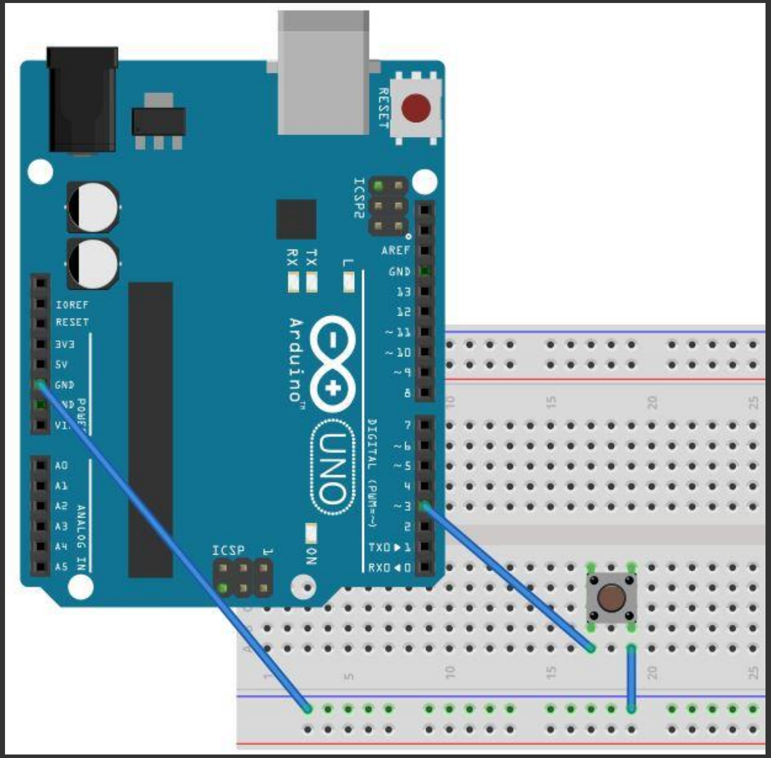




Level 1: "Rolling" the dice with a button (3)



Example: connecting the button



Button Operation

pinMode(Button_Pin, INPUT);
digitalWrite(Button_Pin,HIGH);

The Button_Pin PIN is set to 5V level. When the button is pressed, the PIN level is instantly set to 0V.

Level 1: “Rolling” the dice with a button (4)

1

```
const int Button_Pin = 3;            //Button Pin
const int Debounce_Delay = 50;   //Wait for Button stability
const int roll_delay = 1000;         //Delay between rolls
const int LED_Pins[] = {5,7,4,6};   //LED (dice) pins
const int dice[6][4] =   {{HIGH,LOW,LOW,LOW},//1
                           {LOW,HIGH,LOW,LOW},  //2
                           {HIGH,HIGH,LOW,LOW}, //3
                           {LOW,HIGH,HIGH,LOW}, //4
                           {HIGH,HIGH,HIGH,LOW},//5
                           {LOW,HIGH,HIGH,HIGH}};//6
//Previous button state
int Prev_Button_State = HIGH;
```

2

```
void setup()
{
for(int i=0;i<4;i++)
 {          //Set dice pins as outputs
            pinMode(LED_Pins[i], OUTPUT);
 }
 //Initialize button pin
 pinMode(Button_Pin, INPUT);
 digitalWrite(Button_Pin,HIGH);
 //Initialize random number generator
 randomSeed(analogRead(A0));
}
```

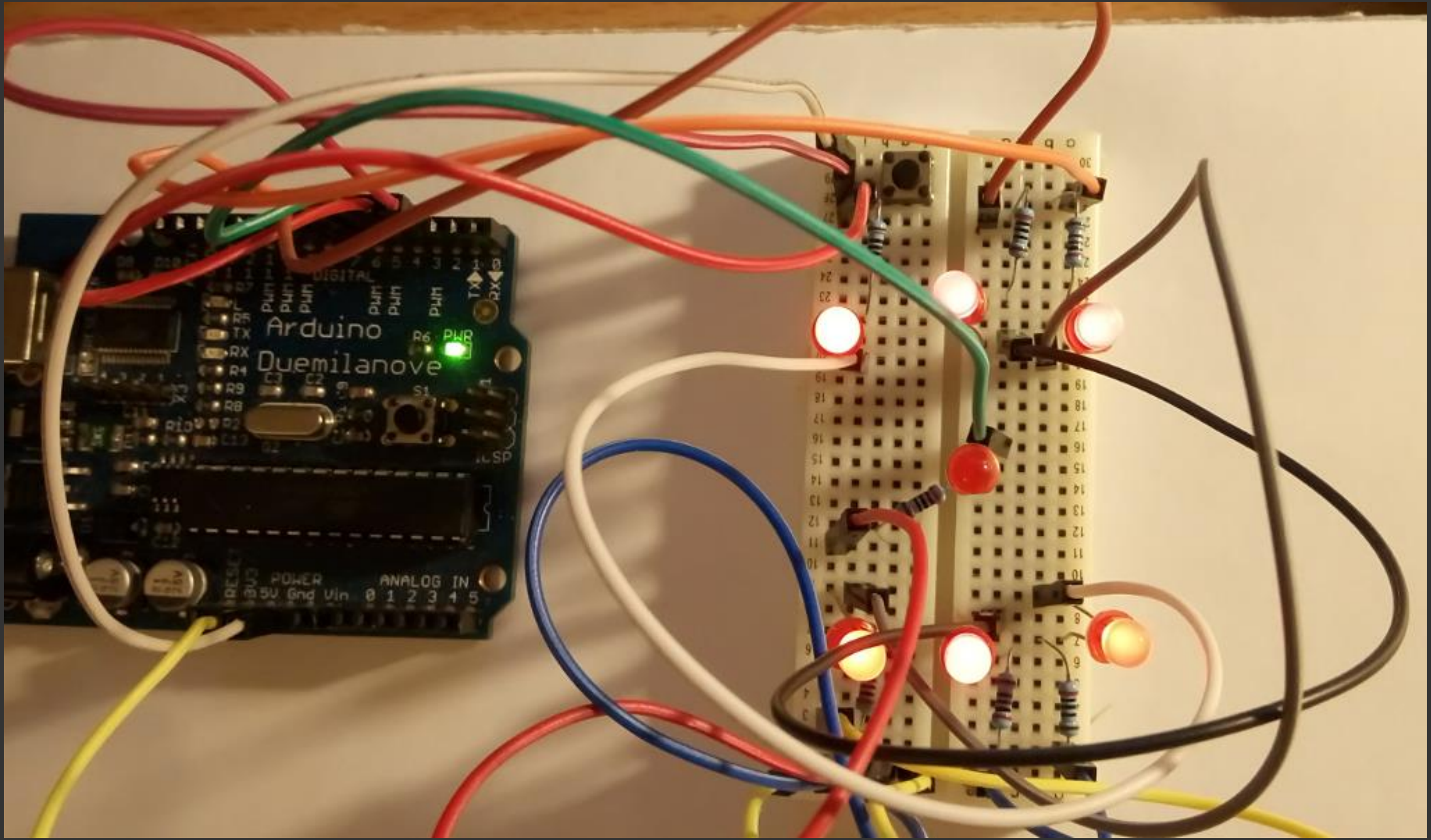
3

```
void loop()
{//Read button state
 int Button_State = digitalRead(Button_Pin);
 if(Button_State != Prev_Button_State)
            delay(Debounce_Delay);
 if((Button_State==LOW)&&(Prev_Button_State==HIGH))
            roll();
 Prev_Button_State = Button_State;
}
```

4

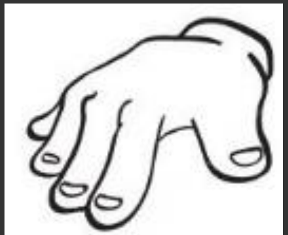
```
void roll() //Rolling the dice!
{
 int i;
 int result = random(1,7);            //Random number 1 - 6
 for(i=0;i<4;i++)//LEDs off
            digitalWrite(LED_Pins[i], LOW);
delay(roll_delay);                    //Delay before new result
 for (i=0;i<4;i++)                    //Display on dice
            digitalWrite(LED_Pins[i],dice[result-1][i]);
}
```

Level 1: “Rolling” the dice with a button (5)

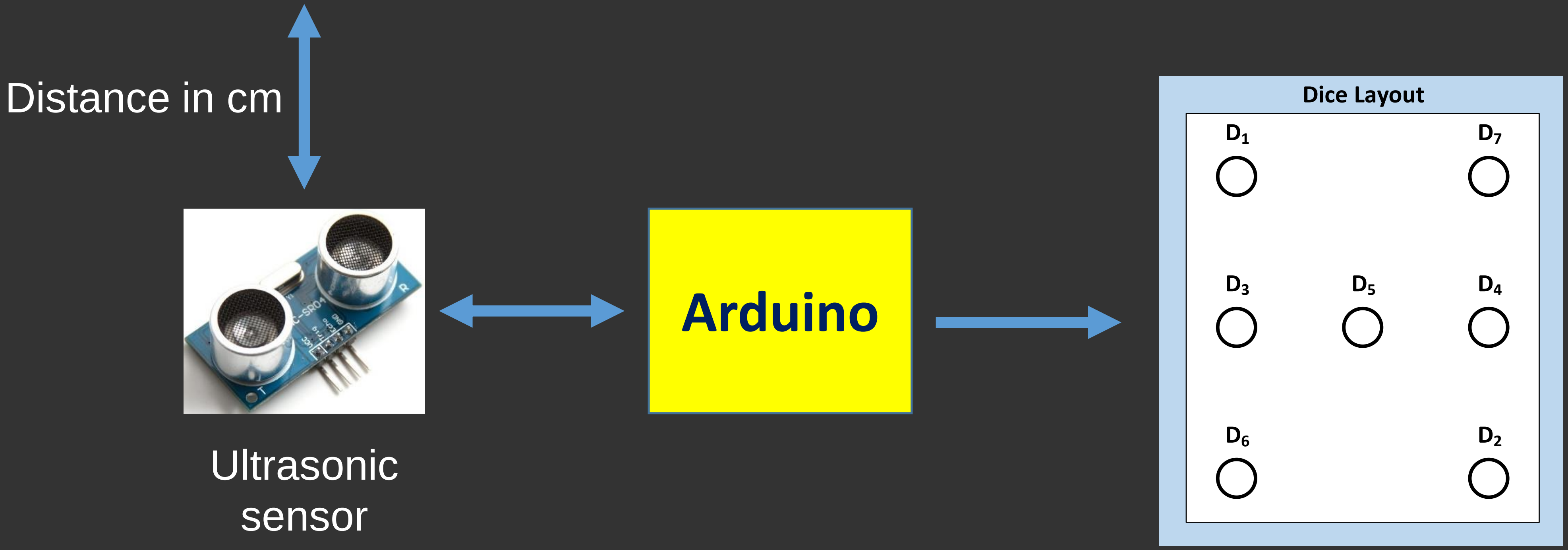


Example: number six on the dice

Level 2: Distance dice “Rolling” using an ultrasonic sensor (1)

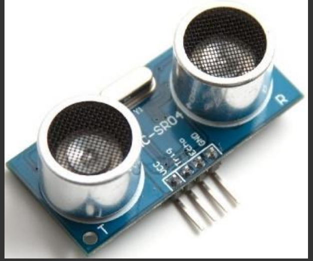


If the distance of an obstacle is less than 5 cm, then a new dice rolling is performed



Level 2: Distance dice "Rolling" using an ultrasonic sensor (2)

Ultrasonic sensor pins



Vcc, Gnd: Power supply
Echo: sensor response
Trig: activate obstacle detection

Obstacle detection procedure (measuring distance)

1. Activate sensor (send triggering), Trig pin
2. Read sensor response (Echo pin)
3. Calculate distance

Level 2: Distance dice “Rolling” using an ultrasonic sensor (3)

1

```
const int Echo_Pin = 11;              //Response pin
const int Trigger_Pin = 12;           //Trigger pin
const int roll_delay = 1000;
const int LED_Pins[] = {5,7,4,6};
const int dice[6][4] = {{HIGH,LOW,LOW,LOW},//1
                        {LOW,HIGH,LOW,LOW},//2
                        {HIGH,HIGH,LOW,LOW},//3
                        {LOW,HIGH,HIGH,LOW},//4
                        {HIGH,HIGH,HIGH,LOW},//5
                        {LOW,HIGH,HIGH,HIGH}};//6
```

2

```
void setup()
{
 for(int i=0;i<4;i++)
    pinMode(LED_Pins[i], OUTPUT);
 pinMode(Trigger_Pin, OUTPUT);
 pinMode(Echo_Pin, INPUT);
 randomSeed(analogRead(A0));
}
```

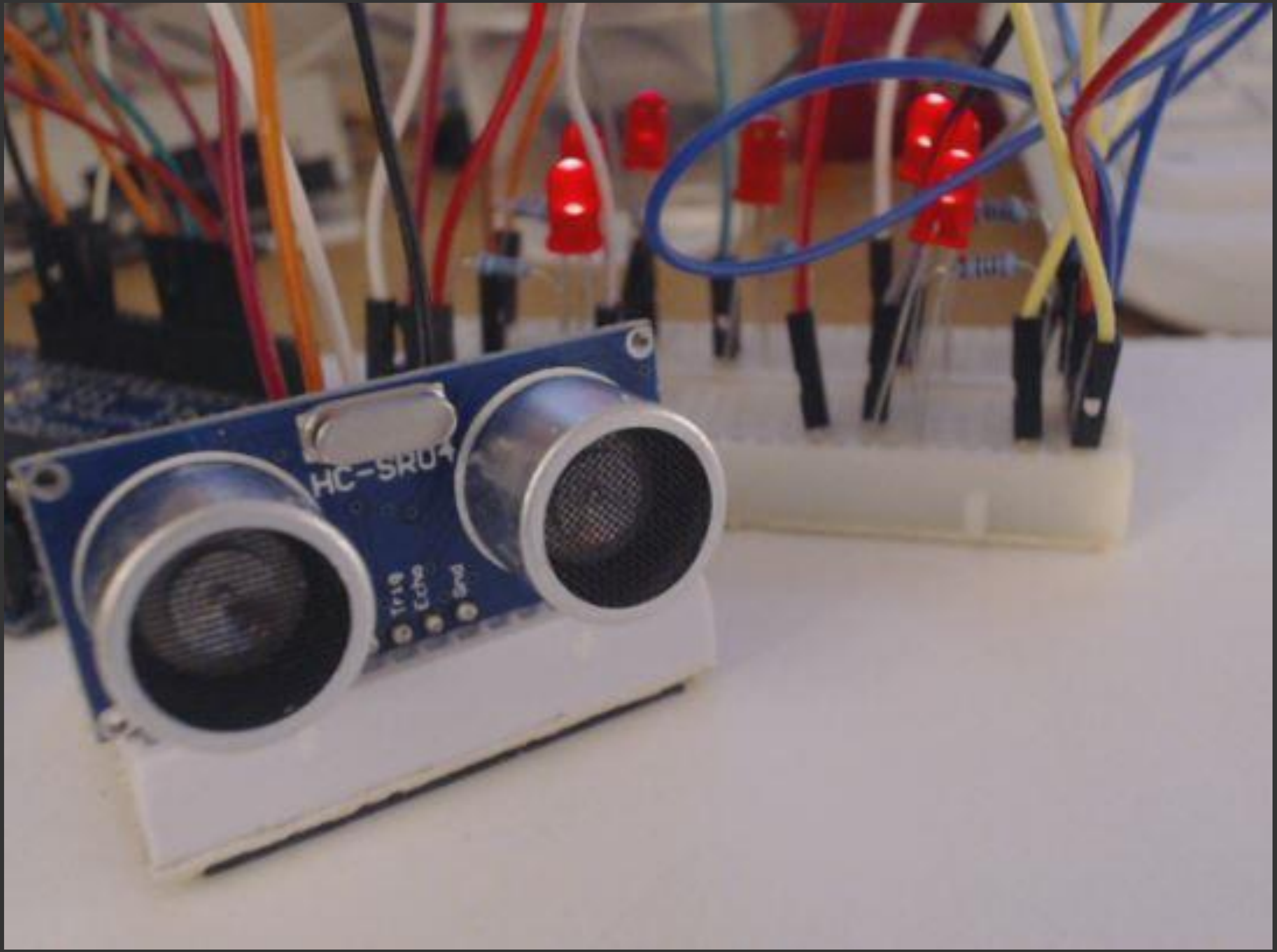
3

```
void loop()
{
 long duration;//Response pulse duration
 float distance; //Real distance
 //Detect obstacle
 digitalWrite(Trigger_Pin,HIGH);
 delayMicroseconds(11);
 digitalWrite(Trigger_Pin,LOW);

duration = pulseIn(Echo_Pin, HIGH);
 distance = 0.034*duration/2;
 if(distance < 5) //Roll the dice for obstacle distance < 5cm
   roll();
  else
  delay(200);
}
```

4

```
void roll()
{
 int i;
 int result = random(1,7);
 for(i=0;i<4;i++)
    digitalWrite(LED_Pins[i], LOW);
 delay(roll_delay);
 for (i=0;i<4;i++)
    digitalWrite(LED_Pins[i],dice[result-1][i]);
}
```

Level 2: Distance dice “Rolling” using an ultrasonic sensor (4)



Thank you!