HIMALAYAN MAKERS GUILD

Foundation Activity 10 Servo Lock Box

CONTENTS AND LEARNING OUTCOMES

Students will,

1. Configure a button as a digital input to an Arduino microcontroller
2. Use digital input and output to make a locking box with a button and servo motor

This activity should take **~1 hour (1.5 hours recommended)** to complete:

MATERIALS AND COSTS PER STUDENT

This activity assumes free access to computers capable of connecting to the Arduino with a USB cable and running two programs:

1. Arduino IDE: https://www.arduino.cc/en/Main/Software
2. BlocklyDuino: https://github.com/BlocklyDuino/BlocklyDuino (or a similar visual programming tool for the Arduino e.g. ArduBlock or S4A)

Both programs can be run in a web browser, or downloaded for offline use. At least one computer and Arduino microcontroller board per three students is recommended.

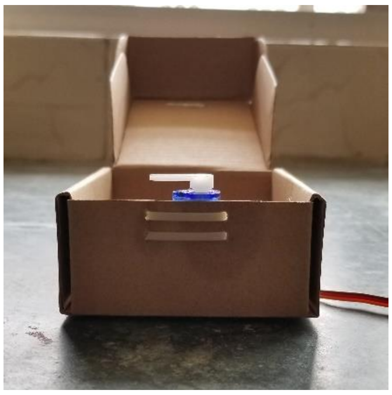
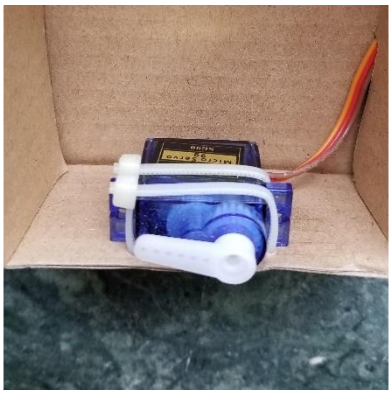
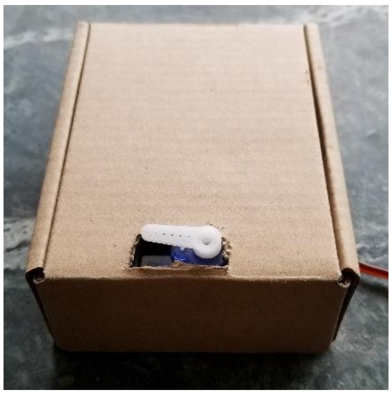
Assuming one kit of parts per student:

Item	**Qty.**	**Cost per Student**[1]	**Expendable**[2]	**Supplier**
Push Button	1	0.02		AliExpress
Resistors, 1k ohm, ¼ W	1	0.01	Y	AliExpress
9V Battery Snap	1	0.16		AliExpress
Jumper cables, MM, 10cm	5	0.10	Y	AliExpress
Breadboard 400 point	1	1.49		AliExpress
Servo SG90	1	1.78		AliExpress
Arduino UNO with cable	1	6.62		AliExpress
9V Ni-Mh 450mAh	1	5.17		AliExpress
Total Cost per Student		**$15.36CAD**		

1. *Currency is CAD, 2017-06-10. Assuming one set of parts per student.*
2. *Likely to be broken or lost during the activity.*

Each student should also get one printed copy of the activity handout. Check the pinout (wire colour and order) of your servo motor, as it may be different from the one used in the instructions below; adjust the lesson and handout accordingly.

The lock box can be made with any closed cardboard box. If the box lid is not one solid piece, tape the top closed and cut 3 edges so that it lifts entirely from a single edge. Position the servo motor on the inside wall of the box, opposite the hinge of the lid, so that the rotor sticks up just above the height of the closed lid. Secure the servo with cable-ties pushed through the cardboard wall to hold the servo above and below its mounting tabs.

Using the functions introduced in Foundation Activity 9, program the Arduino so that the servo motor turns to 90 degrees. Position the plastic arm so that it passes through the opening in the lid at the 90 degree position (unlocked) and secure it to the rotor with the small screw provided or a drop of glue. Then, to open the box, it can be rotated to 0 degrees or 180 degrees, depending on the orientation of the motor in the box.

LESSON

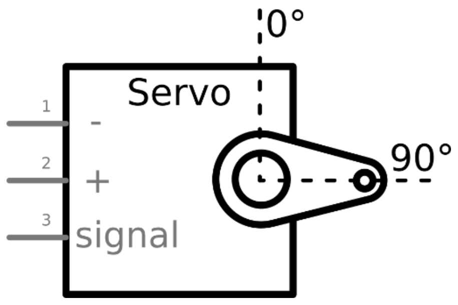
Bold text indicates directions or notes specifically for the instructor.

Prior to starting the lesson, assemble the cardboard lock boxes with servo motors as described in §Materials and Costs per Student. Also complete an example of the lock box with the Arduino UNO and input button. This will be used for demonstration when starting the activity.

ACTIVITY OVERVIEW (10 MINUTES)

In the last activity we used an Arduino microcontroller to make a servo motor rotate between two different angles. Unlike DC motors which spin continuously, a servo motor can be instructed to turn to a specific angle. How many wires does a servo motor have? A: 3 wires, (+) and (-) to power the motor like a DC motor, plus one extra wire for the control signal

Draw the servo diagram on the board:

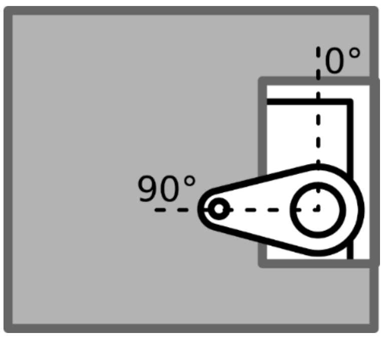


Today, we’re going to:

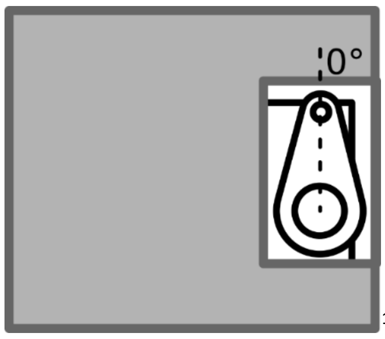
1. Create a box that locks and unlocks using a servo motor
2. Use a button to control whether the box is locked or unlocked

Demonstrate the function of the lock box with a completed example, emphasizing the 3 parts 1. button→ 2. Arduino→3. servo motor.

Box Locked



Box Unlocked



[1]

When the button is not pressed, the motor should turn so the lid of the box cannot be lifted.

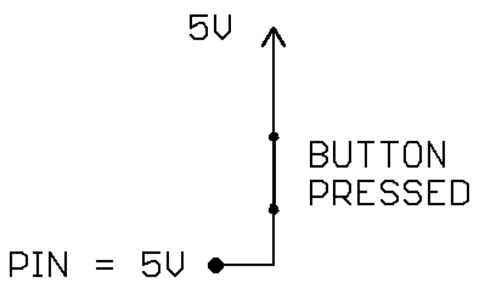
When the button is pressed, the motor should turn to allow the lid to be lifted.

BUTTON AS A DIGITAL INPUT (10 MINUTES)

When turning an LED on and off with the Arduino, we used a digital output pin. Today, we want to read a voltage value with the Arduino using a digital input pin. Like a digital output, a digital input pin on the Arduino reads ~5V as HIGH, and 0V as LOW.

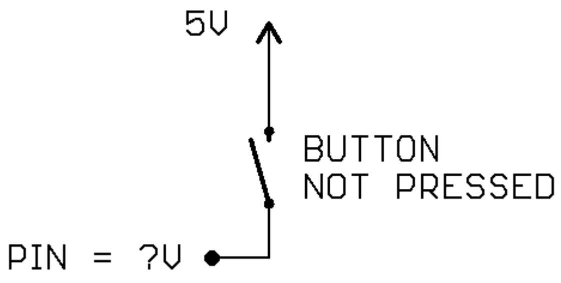
We want to use a button to give the Arduino a digital input signal. What happens if we connect a button directly between an input pin and 5V? **Draw the circuit on the board and ask the students:**

What voltage is the input pin when the button is pressed?



A: The pin is connected directly to 5V, and the Arduino gets a clear HIGH value.

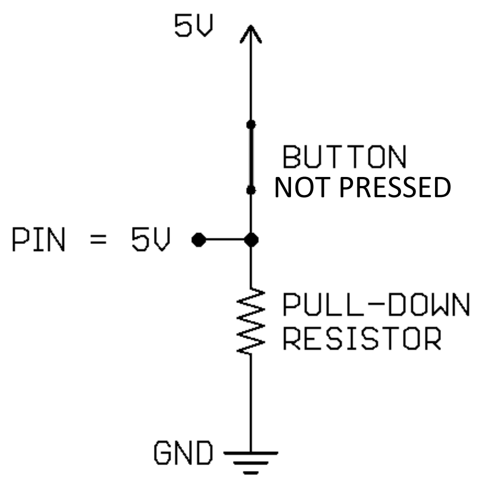
What voltage is the input pin when the button is not pressed?

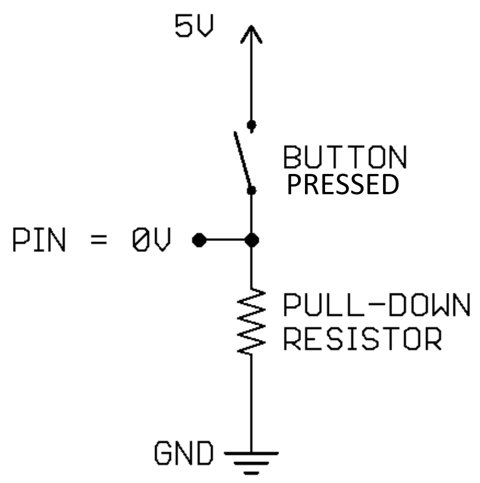


A: The pin was 5V, but now it's not connected to any clear voltage level! It could be read as HIGH or LOW by the Arduino!

[1]Servo motor and Arduino breadboard images from Fritzing

To give the input pin a clear digital value of 5V or 0V from a button, we use a pull-down resistor.





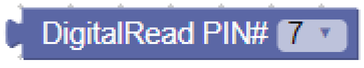
The pin is connected directly to 5V and current flows through the resistor from 5V to GND, giving a clear HIGH digital value.

The resistor pulls the pin voltage down to GND so there is a clear LOW digital value on the pin.

READING DIGITAL INPUT WITH THE ARDUINO (5 MINUTES)

In our program, we will use the DigitalRead function to check if the button is pressed:

DigitalRead is under "Input/Output" in the left side-bar menu in BlocklyDuino. It has one argument, the PIN# that the Arduino should read. This is how the Arduino can check to see if the button is pressed.



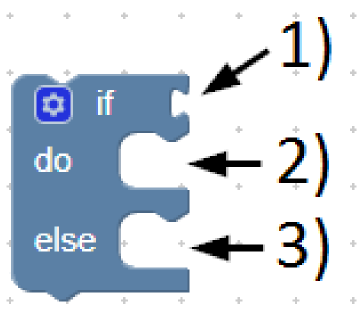
If the pin being read is ~5V, DigitalRead will return a HIGH value to the program. If it is ~0V, it will return a LOW value.

IF-ELSE STATEMENTS WITH THE ARDUINO (10 MINUTES)

Using DigitalRead to understand if the button is pressed or not, we want the Arduino to respond with different instructions depending on the state of the button: If the button is pressed, unlock the box; otherwise, lock the box.

For this we use an if/else statement, which has three parts:

1. **if**: the condition or question the program should respond to.
2. **do:** What to do if the condition is true.
3. **else:** What to do if the condition is false.



So for the lock box,

1. **if:** the button pressed
2. **do:** unlock the box
3. **else:** lock the box

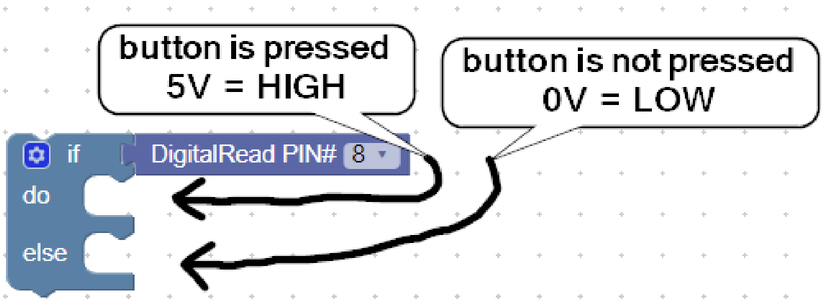
In an if/else statement condition (1.), a HIGH value means true (go to 2.) and reading a LOW value means false (go to 3.).

PROGRAM AND TEST THE LOCK BOX (20 MINUTES)

Combining the DigitalWrite and the if/else statement, our instructions will look like this:

So when the button is pressed, the Arduino will do the instructions that are in "do".

When the button is not pressed, the Arduino will instead do the instructions in "else"

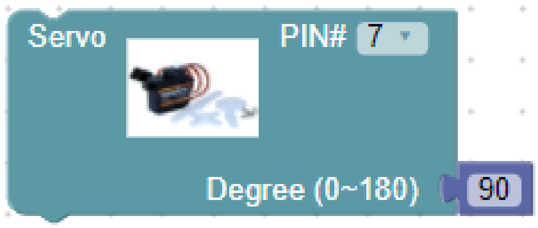


Inside "do" and "else" we will use the Servo-Degree and Delay functions that we used in Foundation Activity 9 to control the servo motor. To review:

Servo – Degree Function

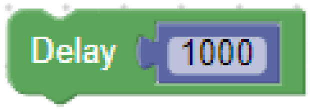
Click "**Servo**" from the left side-bar menu. Select the top block. Set the two arguments:

1) **PIN#**: The output pin that is connected to the "Signal" wire of the motor
2) **Degree**: The degree we want the motor to turn to.



Delay Function

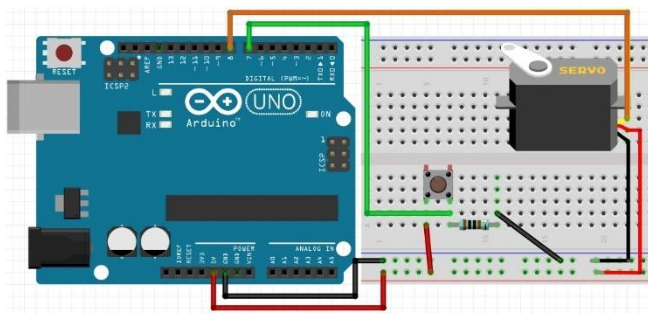
Click "**Control**" from the left side-bar menu. Select the Delay block. The one argument is how many milliseconds the Arduino should wait before continuing on to the next instruction in the program.



Your goal is to connect the button and servo motor to the Arduino, and make a lock box that will open only when the button is pressed. Once you've uploaded and tested your code, try it disconnected from the computer, powered only by a 9V battery plugged into the Arduino.

Note that the unlocked position for your box should be <u>90 degrees</u>, but depending on your box the locked position may be <u>0 degrees or 180 degrees</u>.

Distribute the parts to the students and have them assemble them attach the button and servo motor to the Arduino, then write a program for the lockbox and upload it to the Arduino. When connected with the button and servo, the Arduino setup may look like this:



DEBRIEF DISCUSSION (5 MINUTES)

Encourage a discussion among the students for them to share their thoughts on the activity.

Today we used an Arduino microcontroller board to make a box that locks and unlocks using a servo motor, which is controlled by a button (digital input). Why is this important? What applications does this have?

- Combining both digital inputs and outputs, we can now program the Arduino to respond to outside signals. Today we used a button, but it could also be other types of sensors like temperature or light sensor, or even communication signals from other computers.
- If/Else statements allow us to program the Arduino to follow different instructions depending on certain conditions. Instead of always following the same loop, it can do many different things in one program. This a key concept in programming that can be used in many different types of programming.

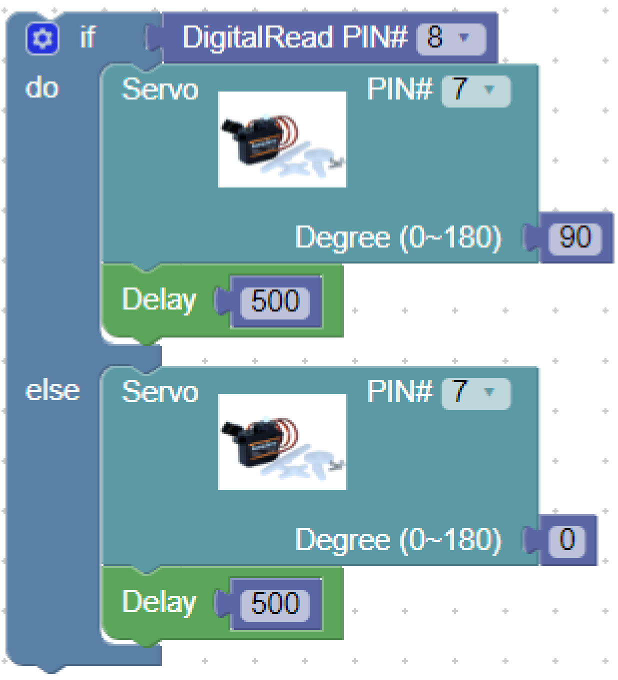
What worked? What didn't work? Why didn't it work? What could we do next, or how could we make the circuit better?

CHALLENGE AND EXPLORE

If a student completes the lesson early, evaluate their understanding by asking them to try the following:

- Change your program to reverse the behavior of the button, so that when it is pressed the button is locked, and when it is not pressed the box is unlocked.

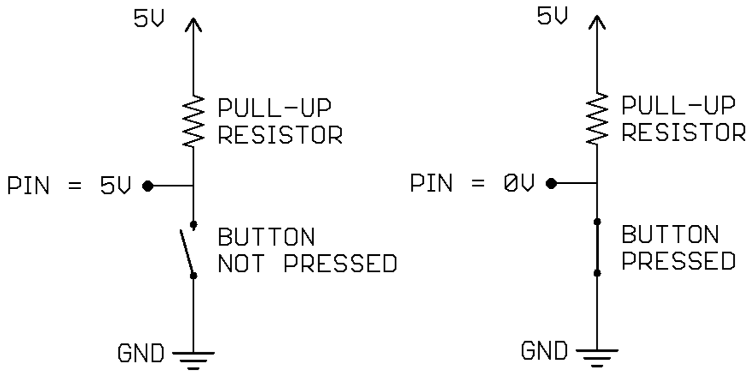
A:



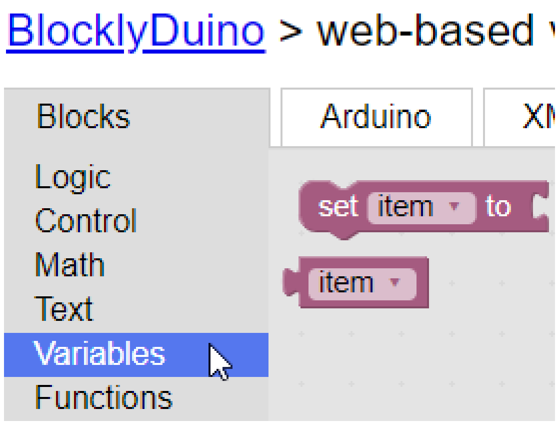
```
if DigitalRead PIN# 8
do   Servo PIN# 7
       Degree (0~180) 90
     Delay 500
else Servo PIN# 7
       Degree (0~180) 0
     Delay 500
```

- Change the button pull-up resistor to a pull-down resistor so that pressing the button makes the digital input pin LOW, and releasing the button makes the digital input pin HIGH. Try it out to see how it changes the behavior of the lock box.

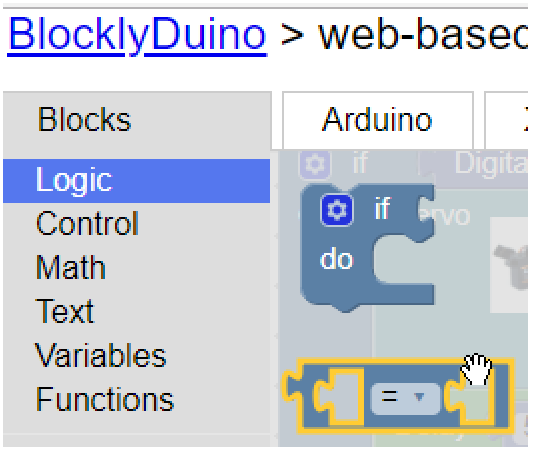
A:



- Change the program so that the state changes between locked and unlocked when pressed? So, if the box is locked, pressing the button will open it, then pressing the button again will return it to being locked. To do this, try using a variable to keep track of the angle of the servo motor. You can make and change variable using the blocks under “Variables” in BlocklyDuino:

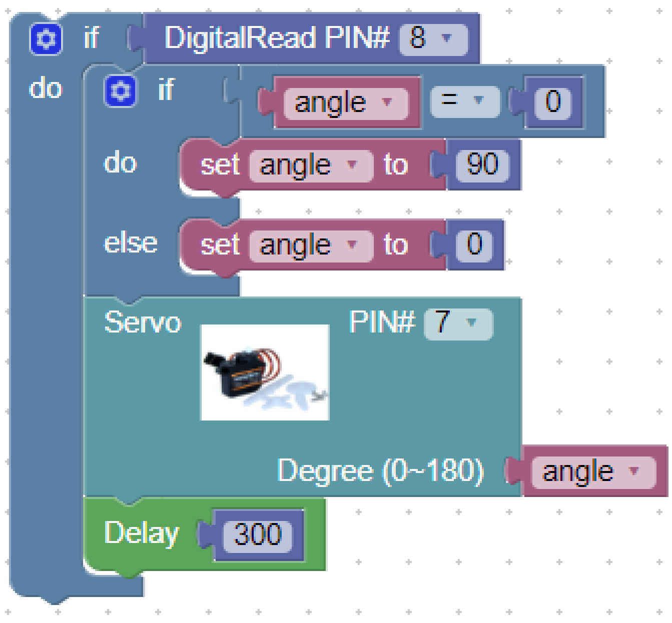


You will also need to use a comparison, to decide what the angle should be. It can be found under “Logic”:



So, when the button is pressed, if the angle is 90 degrees, then change it 180 (or 0, if that is the locked position for your box), else change the angle to 90 degrees. This way it will always switch between 90 and 180 degrees.

A:



```
if DigitalRead PIN# 8
do  if angle = 0
    do   set angle to 90
    else set angle to 0
    Servo PIN# 7
          Degree (0~180) angle
    Delay 300
```

FREQUENTLY ASKED QUESTIONS

- Why is the code not compiling properly?
  - Make sure that all code was deleted from the Arduino IDE before pasting the new code from BlocklyDuino. Also, make sure that all the code was properly copied from BlocklyDuino!
- Why is the code is not uploading successfully?
  - Make sure the Arduino is connected, and that the correct COM port is selected (step 6-8 in §Programming the Arduino)
- Why isn’t my servo motor not moving?
  - A: have you successfully uploaded the code using the USB cable?
  - A: do the pin numbers you used in your code match the pin number where the signal wire of the servo motor is connected and the pin where the button is connected?
  - A: is the (+) wire of the servo motor connected to 5V on the Arduino, and the (-) wire of the servo motor connected to GND on the Arduino?