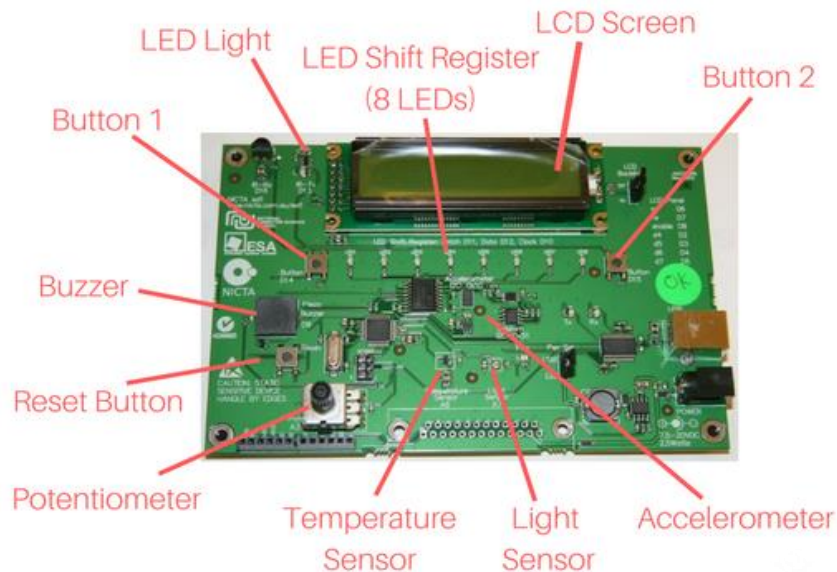


Arduino Attack!

Today we'll be working with **NICTA Ed1 Boards**. They're a special kind of **Arduino board** that comes with all the parts already connected.



1. Flashy Flashy!

Make the LED light flash!

Tasks:

- Make the light flash
- Change how fast the light flashes

1a. Make the light flash!

- 1) Open the file named "Blink.ino"
- 2) Put in the pin number where the question mark is.
- 3) Click the **arrow button** to **upload the code to the board**
- 4) See if the LED light flashes just to the left of the big screen!

1b. Change how fast the light flashes!

- 1) Change **how long the light stays on** by changing the **delay** on the 4th last line of the code (line 12)
- 2) Change **the gap between flashes** by changing the **delay** on the 2nd last line of the code (line 14)

2. Hello World!

Tasks:

- Print "Hello World!" to the LCD Screen
- Print your name on the second line of the screen
- Print the number of seconds since the program started running on the second line of the screen.

2a. Print "Hello World!"

- 1) Open the "LCDHelloWorld.ino" file
- 2) Run the code!
- 3) **Change the text** on line 20 from "<YOUR TEXT HERE>" to "Hello World!"
- 4) Run your code!
- 5) Watch the LCD Screen

2b. Add a second line of text

- 1) **Add an extra line** of code below the line that prints "Hello World!"
- 2) Add code here to **move the LCD screen cursor to the next line.**
This code looks like:

```
lcd.setCursor(0, 1);
```

*0 means the start of the line and the 1 means go to the second line.
(Remember in computer science we start counting from 0. 0...1...2...3)*

- 3) **Print your name** on the second line of the LCD screen.
- 4) Run the code!

2c. Make the screen change

Let's make the screen count up how many seconds the program has been running for.

- 1) **Get rid of the code you added** in part 2b.
- 2) Go to the variable declaration part of the code at the top. **Create an int variable to store the time**, set it to 0 to start with. Use this line of code:

```
int timer = 0;
```

- 3) Go to the **loop** part of the code. The code in this loop will get run over and over again.
- 4) Add code in here that will **move the LCD screen cursor** to the start of the second line every time the loop runs. *Remember how you moved the cursor to the start of the second line above?*
- 5) On the next line add code that will **print the number of seconds** since the program started. **Use lcd.print again together with the timer variable.**
- 6) On the next line **add a delay of 1000ms** (That's one second!) . Use this line of code:
`delay(1000);`
- 7) **Increment your counter.** That means **overwrite** the variable *counter*, **by adding 1** to the current value of counter.
- 8) **Run the code!**

3. Eight LEDs! Eight Times the Fun!

The LED shift register contains 8 LEDs, but unlike the LED we used in *Flashy Flashy* you can't just turn any individual LED on and off. All the LEDs work together, the way we get lights to turn on is by giving the shift register a number, it will then display it in binary using the 8 lights.

Tasks:

- Find out what the numbers 5, 27, 173 and 255 are in binary
- Turn binary numbers into Base-10
- Create your own light pattern and make it appear on your board
- Make it look like lights are moving

3a. Light it up!

Let's see what the LED shift register does. We'll use it to find out what some numbers are in binary.





- 1) **Open the file** named LEDShiftRegister.ino
- 2) **Run the code**, the program will turn the lights that display the number 53 in binary. *The lights light up like 00110101, where 0 is light off and 1 is light on. In binary we drop the leading 0's (just like when we write numbers normally, which is called Base-10),so it becomes 110101.*
- 3) **Find the line** in the code that says `displayBinary(53)`
- 4) **Change the number** 53 in the `displayBinary` code to find out what the following numbers are in binary to fill in the table:

	8 bit binary	Shortened
53	00110101	110101
5		
27		
173		
255		

Binary Facts

Binary numbers are like a set of light switches, you get to decide which columns you want to turn on. Turn on the light switches that you need to added up to make a number.

To make the number eighty-three what numbers do we want to tick to include in our sum?

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One
							

add up the columns that have ticks in them

Sixty-four + sixteen + two + one = Eighty-three

1's and 0's

We can do what we did with ticks with ones and zeros by replacing ticks with 1 and blanks with 0.

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One
0	1	0	1	0	0	1	1

The binary number for 83 is 01010011. We can drop the first 0 to make it 101001.

3b. Binary numbers to Base 10

Steps:

1. **Use the tables** to see the binary number in 8 pieces
2. **Add up** the columns with 1 in them to turn it into a Base-10 number
3. Put your answer into this code from step 3a:
`displayBinary(<your number>).`
4. **See if the lights match** the pattern of 1's in the binary number you started with to check your answer!

Puzzle 1: What is 10011101 in Base-10?

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One
1	0	0	1	1	1	0	1

Add up the columns with 1's

..... + + + + =

Use your LED Shift Register to check your answers!

Puzzle 2: What is 101010 in Base-10?

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One
0	0	1	0	1	0	1	0

Add up the columns with 1's

.....

Use your LED Shift Register to check your answers!

Puzzle 3: What is 1011 in Base-10?

Start by putting numbers in the table.

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One

Then add up the columns with 1's

.....

Use your LED Shift Register to check your answers!

Puzzle 4: Can you make this pattern using your LED Shift Register?



One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One
1	0	0	0	0	1	0	1

Add up the columns with 1's

.....

Puzzle 5: Can you make this pattern using your LED Shift Register?



Start by putting it as 1's and 0's in the table

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One

Add up the columns with 1's

.....

Puzzle 6: Can you make this pattern using your LED Shift Register?



One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One

Add up the columns with 1's

.....

Puzzle 7: Create your own pattern!

Create your own light pattern by colouring in some of the lights on the picture below.



Now figure out how to create that pattern on the NICTA ED1 Board

One-hundred and twenty-eight	Sixty-four	Thirty-Two	Sixteen	Eight	Four	Two	One

Add up the columns with 1's

.....

3c. Moving Lights

Now we know how to make the lights turn on, we can now make it look the lights are moving.



Base-10 = 1

Binary = 1



Binary = 10
Base-10 = 2



Binary = 100
Base-10 = 4








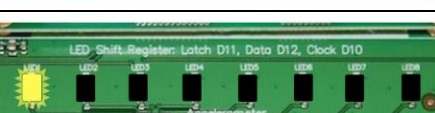
Let's make it look like the light is moving from right to left.

1. **Copy these lines of code** from the loop part of your code.
`displayBinary(<YOUR NUMBER>)`
`delay(<YOUR DELAY>)`
2. **Paste 2 copies** of these below the existing lines.
3. **Change the numbers in displayBinary**, so the lights moves from right to left 2 places. You can get the numbers you need for this from the pictures above.
4. **Run the code** to see the lights move.

3d. More Moving Lights - Less Code

Let's make the lights move all the way to the left side

1. If we kept adding code to add all these lights our programming would get really long! **Get rid of the extra bits of code you copied and pasted in *Moving Lights*.**
2. Complete the table on the next page with the Base-10 numbers that correspond to the lights.

	Binary	Base-10
	1	1
	10	2
	100	4
		
		
		
		
		

3. **What is the pattern** in how the Base-10 number changes when the lights change?

.....

4. **Create an integer type variable called *number***, in the Variable Declaration Section. **Set it to be 1** (the value of the first light on the right). *The value stored in the variable number will be the Base-10 number we want to show on the lights.*

5. In the Loop Part of the code we're going to **change *number*, using the pattern we made in Step 3.**

We're going to find the new value for *number* based on what it used to be.

Finish this line of code:

`number = number ♥ ☆`

Replace the ♥ symbol with plus ,minus, multiply or divide. (the symbols +, -, *, /)

Replace the ☆ symbol with a number

Hint: Remember how we updated the counter in HelloWorld 2.c? It's a little bit like that!

6. **Run your code!** Does your light move all the way across from left to right?
7. **BONUS.** Add code to **make the lights go back the other way.** You'll need to use an if statement to figure out when the code has gotten to the edge.

4. Button Bonanza!

We can use a button to only make things happen some of the time!

Tasks:

- Create a light that turns on while the button is held down
- Switch the light so the light turns off when the button is held down
- Twice the Buttons, Twice the fun. Use the screen to display button presses.
- BONUS: Toggle the light

4a. Light Switch

Let's make a light that turns on while the button is held down and turn off when it is released.

1. **Open the file** called *lightSwitch.ino*, we've already setup the pins for you.
2. **Create an integer variable called *buttonState*** in the variable declaration section. This stores the button state, **set it to *HIGH*** to start with.
(When the the button is up, it's *HIGH*. When the button is pressed it's *LOW*.)
3. In the loop() section **add this line of code** which checks if the button is pushed or not:

```
buttonState = digitalRead(buttonPin);
```

digitalRead finds out if the button is pushed (*LOW*) or not (*HIGH*) and stores it in the *buttonState* variable.
4. Find the if statement we've set up for you. **Inside the if statement add this code to turn on the light when the button is pushed:**

```
digitalWrite(ledPin, HIGH);
```
5. **Run the code and press the button!** Does the light turn on? Does it never turn off?
6. Add an **else statement** after the if statement
7. Inside the else statement **tell the light to turn off when the button is not pressed.** You turn off the light with this code:

```
digitalWrite(ledPin, LOW);
```
8. **Run the code!**

4b. Switch the switch

1. Change what happens inside the if statement in your code from Light Switch, so the light is on **unless** the button is pressed.
Button pushed -> light off
Button releases -> light on
2. **Run the code**, play with it until it works!

4c. Buttons and screens!

We have 2 buttons to use and a screen! Let's print "Left" on the screen when the left button is pushed and write "Right" on the screen when the right button is pushed. If neither button is pushed have a blank screen.

1. **Open the file** called "buttonsAndScreens.ino", all the variables are set up like they were previously for the button (but now we have 2 buttons, buttonPin1 and buttonPin2) and we've set up the LCD screen for you.
2. Inside the Loop Section **move the LCD cursor** to the start of the top line of the screen
3. Let's start with the **left button**.
 - a. Write an **if statement that checks if the left button is pushed**, using button1State.
 - b. After you clear the screen **print "Left" to the screen**. *Go back to your code from "Hello World" if you need to remember how to print.*
 - c. **Run the code and press the button!** Does it print left? Does it never go away?
4. Let's **make the screen blank** when the button is released.
 - a. **Add an else statement** (*Remember you don't need a condition for an else statement*)
 - b. Inside the else statement **clear the screen**.
Remember you do this with lcd.clear()
 - c. **Run the code and press the button!** Does it print left? Does the screen clear when the button is released.
5. Let's take care of the **right button!**
 - a. In between the if statement and the else statement **add an else if** statement to **check if the right button is pushed**. *Don't forget else if needs a condition.*
 - b. **Inside the else if statement** repeat what you did for printing left, but **print right** this time.
 - c. **Run the code!** Does it work?
6. BONUS: **Make it print "BOTH!"** to the screen when both the buttons are held down at the same time;

- a. Change the if statement section to an “else if”
- b. Above this create a new if statement, this will **check if both the left and right buttons are pushed**. Like before check if button1state is LOW and button2state is LOW at the same time.

Join these together in the if statement using &&, which is the way we write “and” in arduino. *Make sure you use extra brackets for the things on each side of the &&*.

- c. Inside the new if statement, print “Both!” to the screen
- d. Run the code!

4d. BONUS: Toggle Trouble!

Let’s make it so that when we press and release the left button once, the light turns on and if you press and release it again it turns off. We call this toggling the light.

1. **Open the toggleTrouble.ino file**, we’ve got a new variable called lightState which stores whether the light is on or not. It’s set to LOW because the light starts off.
2. In the loop() part of the code **update buttonState** with digitalRead like you did for the previous activity.
3. **Write an if statement that checks if the button is pressed**, if it is we need to do something!
4. Inside the if statement create an if statement that **checks if the lights is off** (lightState==LOW).
5. Inside the inner if statement
 - a. **Set the light state to HIGH**
 - b. **Turn on the light** with `digitalWrite(ledPin, lightState);`
6. After the inner if statement, but inside the outer if statement add an else statement
7. Inside the else statement:
 - a. **Set the light state to LOW**
 - b. **Turn off the light**
8. After the else statement closes off, **put a delay of 200 milliseconds**. If we don’t our program runs to too fast and might turn the light on and off faster than we can push the button.

5. Did you hear that?

Tasks:

- Create a musical door bell
- Make a musical instrument

5a. Door Bell

1. **Open the file doorBell.ino.** We've already set up all the pins for the buttons and the buzzer for you
2. **Go to the if statement**
3. Inside the if statement **add this code to make a note play:**

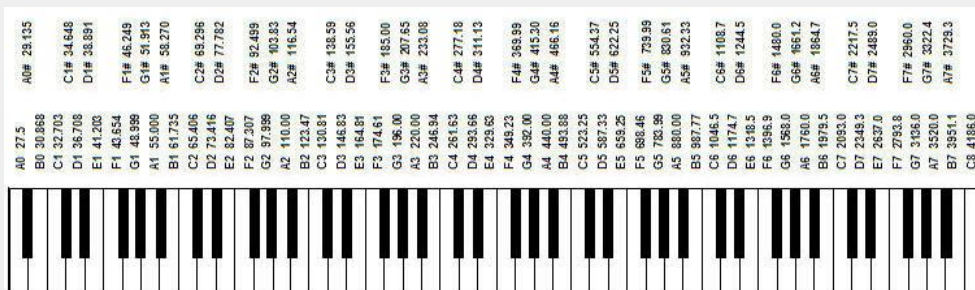
```
tone(piezoBuzzerPin, NOTE_D3, noteDuration);
delay(noteDuration);
```

The note duration is a variable we've defined at the top, is just how long you want the note to play for in milliseconds. **We need to have the same note duration in the tone line as we have in the delay line.**

4. **Run your code**

Music facts

D3 is just part of a naming convention we use for the musical notes. D3 is the the note D in the 3rd octave. If we wanted D sharp (D#) in the 3rd octave we would write DS3. Middle C is called C4.



We have all the notes from B0 to DS8 preloaded into your code.

That's what the #include "pitches.h" does. If you open the "pitches.h" file you'll see all the notes defined by the frequency of their pitch.

5. Copy and paste a copy of the code from above just below the original code. Change the note that plays by **changing NOTE_D3 to another note**.
6. **Add more copies of the lines above** and change the notes and note durations for the different notes **to make a song**. You can find the notes you need for some songs on the GPN website for today.

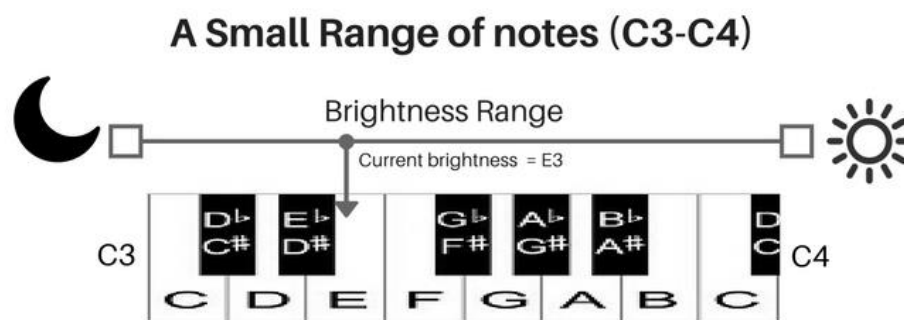
5b. Musical Instrument Magic!

Let's play a magical musical instrument that we control without touching it!

1. **Open the file** called lightTheremin.ino
2. At the top of the file there is a question mark where the Light Sensor pin number should be, **fill in the Light Sensor Pin number** and **Buzzer pin number**. Look at your board to find the number of the light sensor.
Hint: You don't need to include the letter at the start of the number
3. **Run the code** and **follow the calibration instructions** by covering the light sensor when it tells you to.
4. When the calibration is finished **hold down the left button** and **make music with the light sensor**.
5. **Scroll to the bottom** of your file and **find the line of code** near the bottom of the file is a line of code that says this:

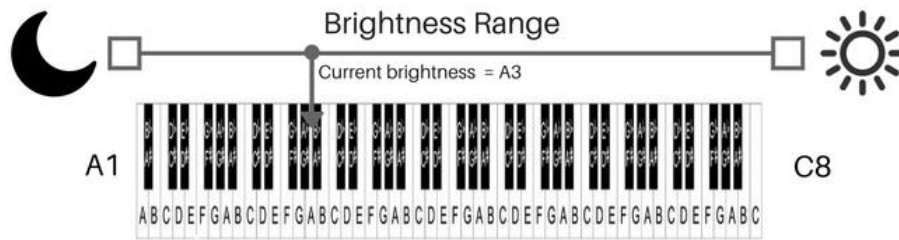
```
int pitch = map(sensorValue, sensorLow, sensorHigh, NOTE_G3, NOTE_B6);
```

Above we have set our lowest note to be G3 and the highest note to be B6. But we can have any range of notes we want by changing these to different notes.



Only a few musical notes stretched out over the range.
Makes notes easy to find and to hold. But you get less notes.

A Big Range of notes (A1-C8)



Lots of musical notes squished into the range.
Makes notes hard to find because it's fine.

The map function works out which note to play, by comparing the current amount of light to the minimum and maximum amounts of light (that it learnt in the calibration).

It then works out the corresponding note compared to the minimum and maximum notes we set for our instrument.

6. **Change the range of notes** to a big one or a small one and see how your light theremin plays differently.