Introduction

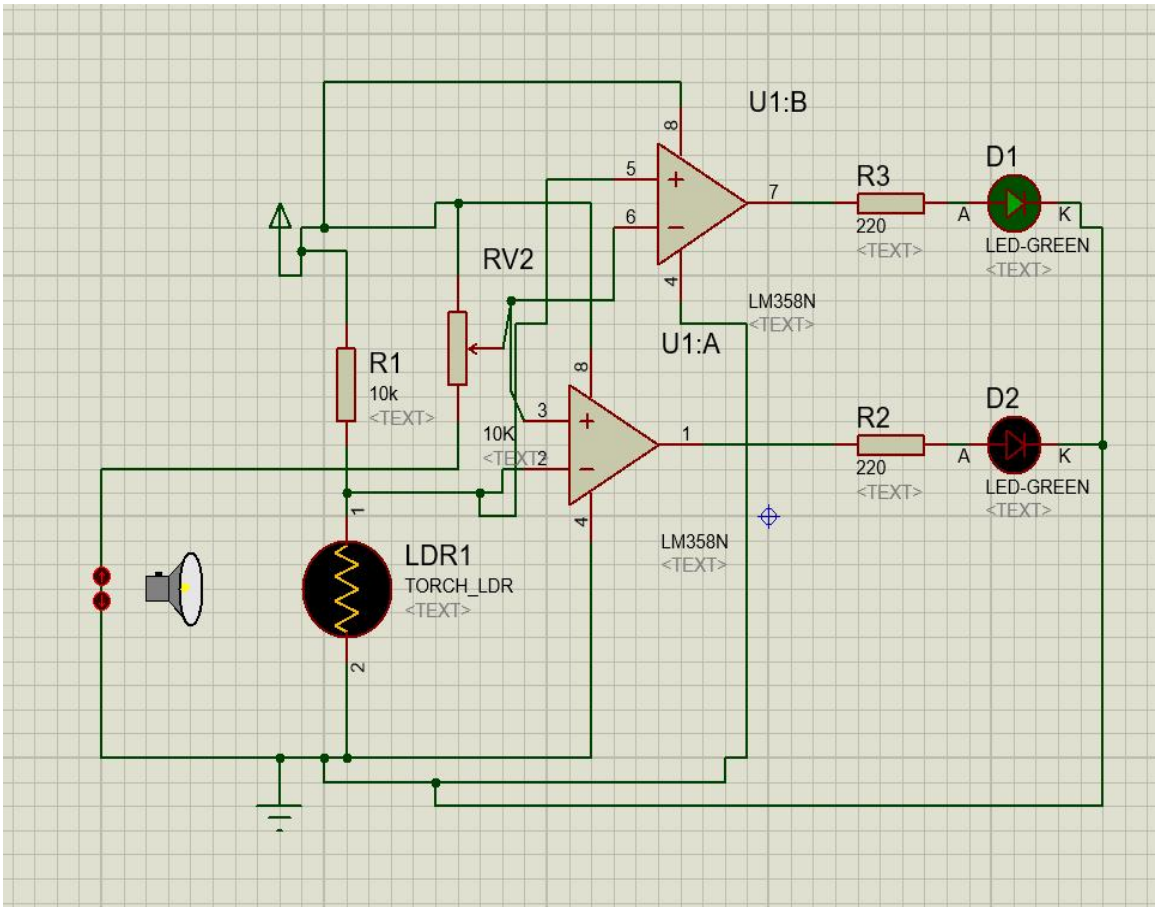
The goal here is to create our own sensor using operational amplifiers. This is to know how sensors are made and how important operational amplifiers are in creating a circuit especially if it involves sensor. In this activity I will create a DARK sensor.

The twist now makes it more exciting as we have to use the sensor using an ATmega328 microcontroller. Our aim is to write a program in assembly language which reads the input and output data of our sensor. This guide will show you how to write assembly code for an Arduino microcontroller that reads a Light Dependent Resistor (LDR) sensor and then produce results. LDR sensors are widely used in many applications where there is need to detect light levels. These are used in automatic lighting systems, light meters and other devices that respond to changes in ambient light. The results can be achieved by utilizing an ATmega328 microcontroller together with LDR sensor making a real-time system that reacts on different illumination changes.

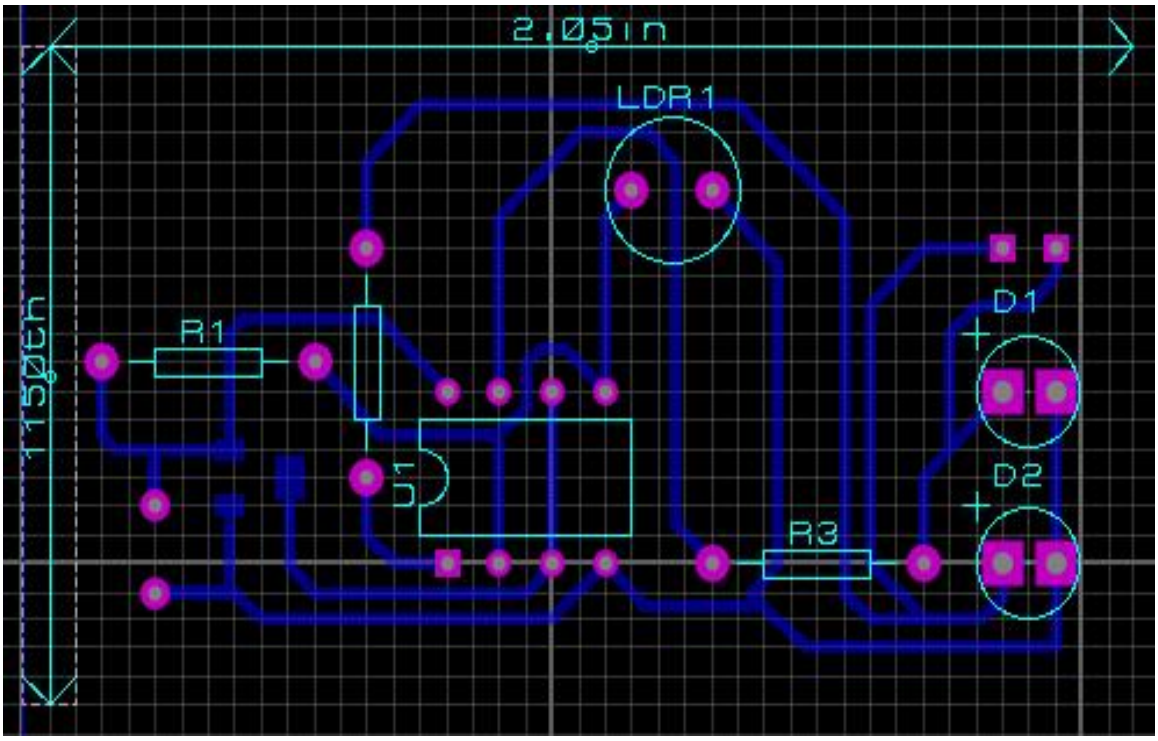
The setup starts from configuring the pins of the microcontroller, initializing relevant registers so as to correctly read the voltage coming from my sensor and receive it to the Arduino and set the output to another pin into an LED that will indicate that it sense a voltage.

In my project, my application is a PARKING LOT. Using it will indicate if that certain spot is already occupied or not with an LED as an indicator.

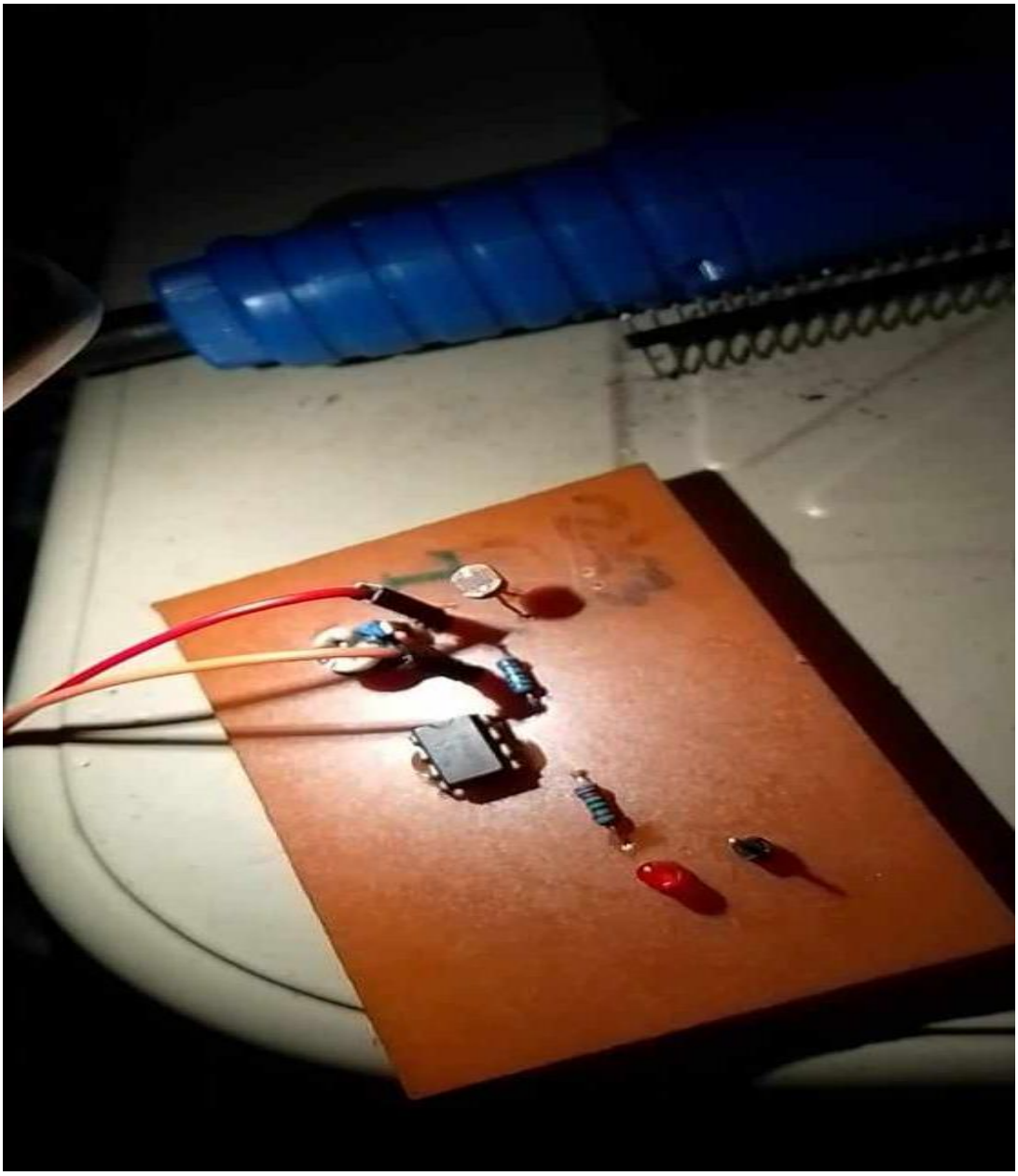
LOGIC DIAGRAM



PCB LAYOUT



PIN CONNECTION



Arduino code to assembly Language

.INO

```
extern "C"

{
  void start();
  void darkF();
}
void setup()
{
  start();
}
void loop()
{
  darkF();
}
```

.S

```
.global start
.global darkF

.equ DARKsensor_PIN, 3
.equ LED_PIN, 4

.equ DDRD, 0x0A
.equ PORTD, 0x0B
.equ PIND, 0x09

start:

    sbi PORTD, DARKsensor_PIN
    sbi DDRD, LED_PIN
darkF:
    sbis PIND, DARKsensor_PIN
    rjmp DARKsensor_PIN_low_df
    sbi PORTD, LED_PIN
    ret

DARKsensor_PIN_low_df:
    cbi PORTD, LED_PIN
    ret
```

How it works?

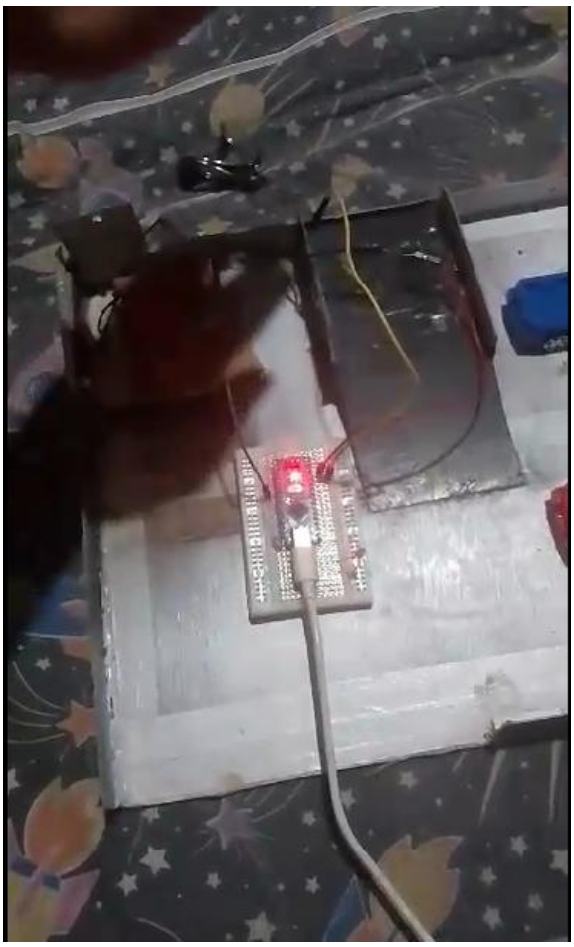
First I make another extension of pin from my sensor and put it to the breadboard which is connected to the pin I set for my Arduino which is exactly at D3 and the output is to D4 and it will be the indicator of the functionality of my sensor.

In my code I'm still using a .INO or C++ code but it is only for the starting and the loop or just for my calling function. In my .S or assembly code will be the process and my main code. The start label marks the beginning of the main program, The darkF label marks the beginning of a function that checks the state of the DARKsensor_PIN and controls the LED_PIN accordingly.

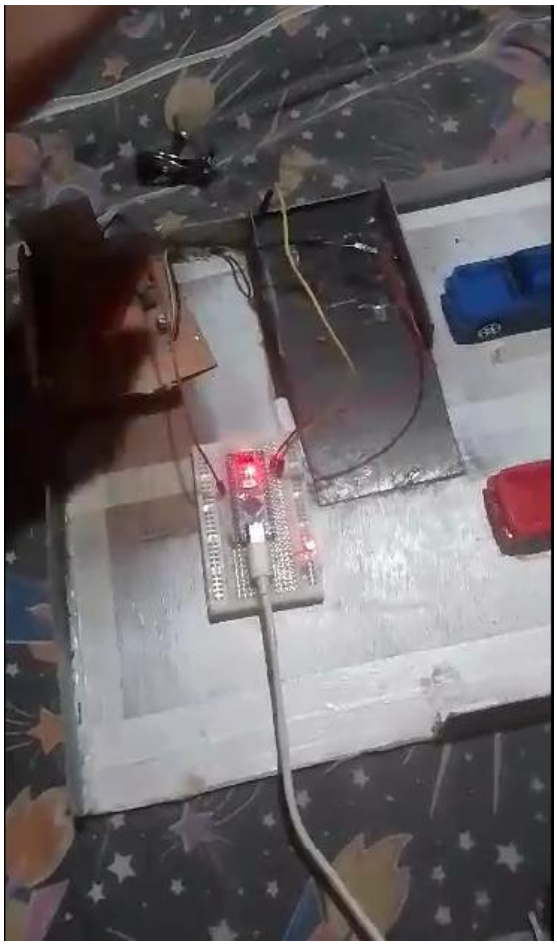
After the Arduino received a pulse from the sensor, it reads if it is high or low and store it in data register. If it is HIGH it skip to the next instruction and the LED that indicate if the slot is occupied will turn on which is RED. If there is no shadow detected it will stay into green. The 'DARKsensor_PIN_low_df' label marks the section where if the 'DARKsensor_PIN' is low, the 'CBI' (Clear Bit in I/O Register) instruction sets the LED_PIN low, turning the LED off. Finally, the 'ret' instruction returns from the function.

The code sets up a digital input pin for a dark sensor and an output pin for an LED. It continuously checks the sensor's state and turns the LED on if the sensor detects darkness (indicated by the sensor pin being high) and off if there is no darkness (sensor pin low).

SIMULATION

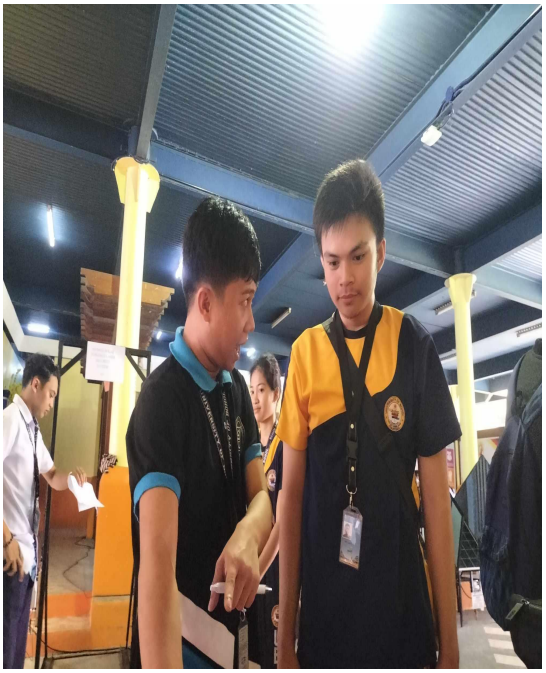


OUTPUT = 0 (LOW)



OUTPUT = 1 (HIGH)

Technocraft display simulations



Project summsry/ hurdles

This project has proved to be a great hurdle for me. The push made me feel like I will not going to pass. At the beginning of us being given this assignment, we had no idea of what we were supposed to do or how to go about it. The first challenge was to convert Arduino code into assembly language that seemed difficult at that time. We were afraid of how can we turn Arduino code into assembly.

We were overwhelmed by the difficulty level at first. Converting code from one language to another, especially assembly language, is more than just translating commands. It involves understanding hardware interactions and optimizing the code for efficiency. We did not have experience in that particular kind of programming before.

Despite these hurdles, I decided to focus on understanding the project deeply and how each part of the code interacted with the sensor. Through persistent effort and determination, I was able to develop an output that correctly interfaced with my sensor. This process involved a lot of trial and error, learning from each mistake, and gradually piecing together a functional program.

Hoping we will pass this subject as we do our best for understanding and study for it. Thanks to my classmate and BSCOMPE students who supporting everyone and make this project successful.

GOOD LUCK EVERYONE!

Curriculum Vitae

Name: John Michael D. Conarco

Age: 25 Years Old

Date of Birth: May 24, 1999

Status: Single

Address: Tupas, Antequera, Bohol

Mobile: 09276934384

Email: jmdconarco@universityofbohol.edu.ph

Motto: *You are never too old to set another goal or to dream a new dream.*

Portfolio: https://michael-2su.pages.dev/