

Lab 8

ADC LAB:

INFRARED PROXIMITY SENSOR

Contents

1	Introduction.....	1
1.1	Lab overview.....	1
2	Requirements	1
3	Details	2
3.1	Hardware	3
3.1.1	Circuit Assembly.....	3
3.2	Software	4
4	Procedure	5
4.1	Basic Steps	5
4.2	Analysis	5
4.2.1	Ambient Light Analysis	5
4.2.2	Time Delay Analysis.....	5
4.2.3	Range Analysis.....	6
4.2.4	Calibration	6

1 Introduction

1.1 Lab overview

For this project, you will create a device that uses reflected infrared (IR) light to detect if an object is nearby.

2 Requirements

In this lab, we will be using the following hardware and software:

- **Keil μ Vision5 MDK IDE**
 - Please see the included [Getting Started with Keil guide](#) on how to download and install Keil.
- **STM32 Nucleo-L552ZE-Q**
 - For more information, click [here](#).
- **Logic Analyzer or Oscilloscope**
 - Required to monitor the interrupt signals
- **IR emitter (LED)**
- **IR detector (photo transistor)**
- **Smattering of wires**
- **RGB LED**
- **100 ohm resistor, 10k ohm resistor, 3 1k ohm resistor**

3 Details

The proximity sensor uses an IR emitter (LED) and an IR detector (photo transistor) pointing in the same direction to determine if any object is reflecting IR energy from the LED.

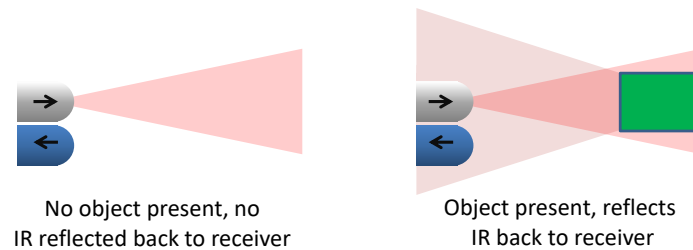


Figure 1. Proximity sensor method of operation.

The proximity sensor works with a combination of hardware and software. Sensing occurs in two steps: First, the software must measure the IR light level (using IR-sensitive photo transistor Q1 and the analog to digital converter) when the IR-emitting LED is **turned off**. Second, the software must measure the IR light level when the IR LED is turned **on**. If the IR brightness level has increased, then there may be an object nearby reflecting the IR from D1 back to Q1. The signal strength is to be indicated by the RGB LED.

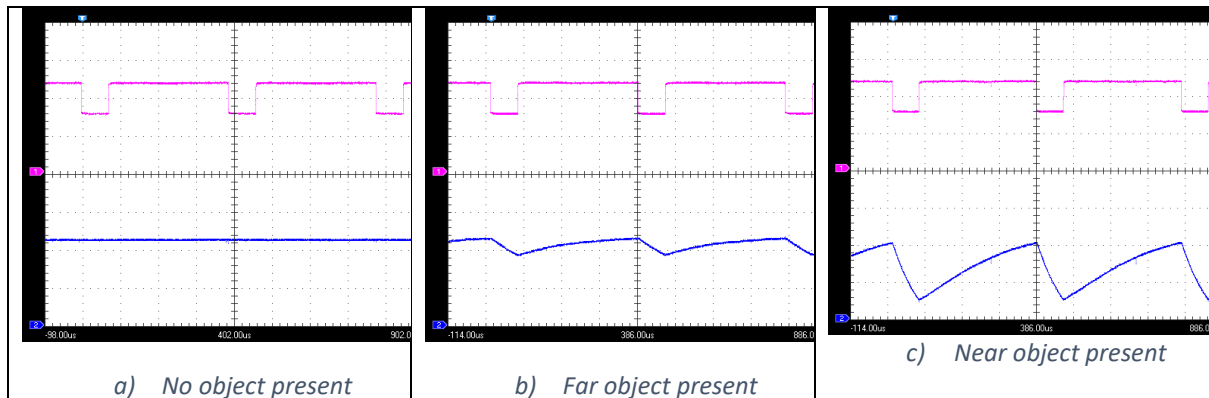


Figure 2. Oscilloscope screenshots showing IR Output Signal (upper trace) and IR Input Signal (lower trace). Your output signal may look different but should be some form of consistent square/rectangular wave if done correctly

As IR energy increases, this increases the conductivity of the photo transistor and lowers the output voltage. Also, note that the photo transistor has a slow response, as shown by the exponential decay curves in Figures 2b and 2c above.

You can verify that the IR LED is turned on by viewing it with a digital camera (e.g. in a cellphone), as these are sensitive to IR light.

3.1 Hardware

Please see the included Nucleo-L552ZE-Q pins legend (NUCLEO_L552ZE_pins.docx) for the pinout of the Arduino-included Zio connectors for CN7, CN8, CN9 and CN10.

3.1.1 Circuit Assembly

Build the circuit on the breadboard as shown below. The RGB LED should be a largely separate circuit from the IR emitter/receiver circuit (outside of they can use the same ground rail). Make sure you verify each resistor value and make the appropriate connections and that you use 3.3 V for Vdd as specified in the table later on. Also, try to position your IR emitter/receiver similarly to those shown in the image (fairly close together and both pointing up)

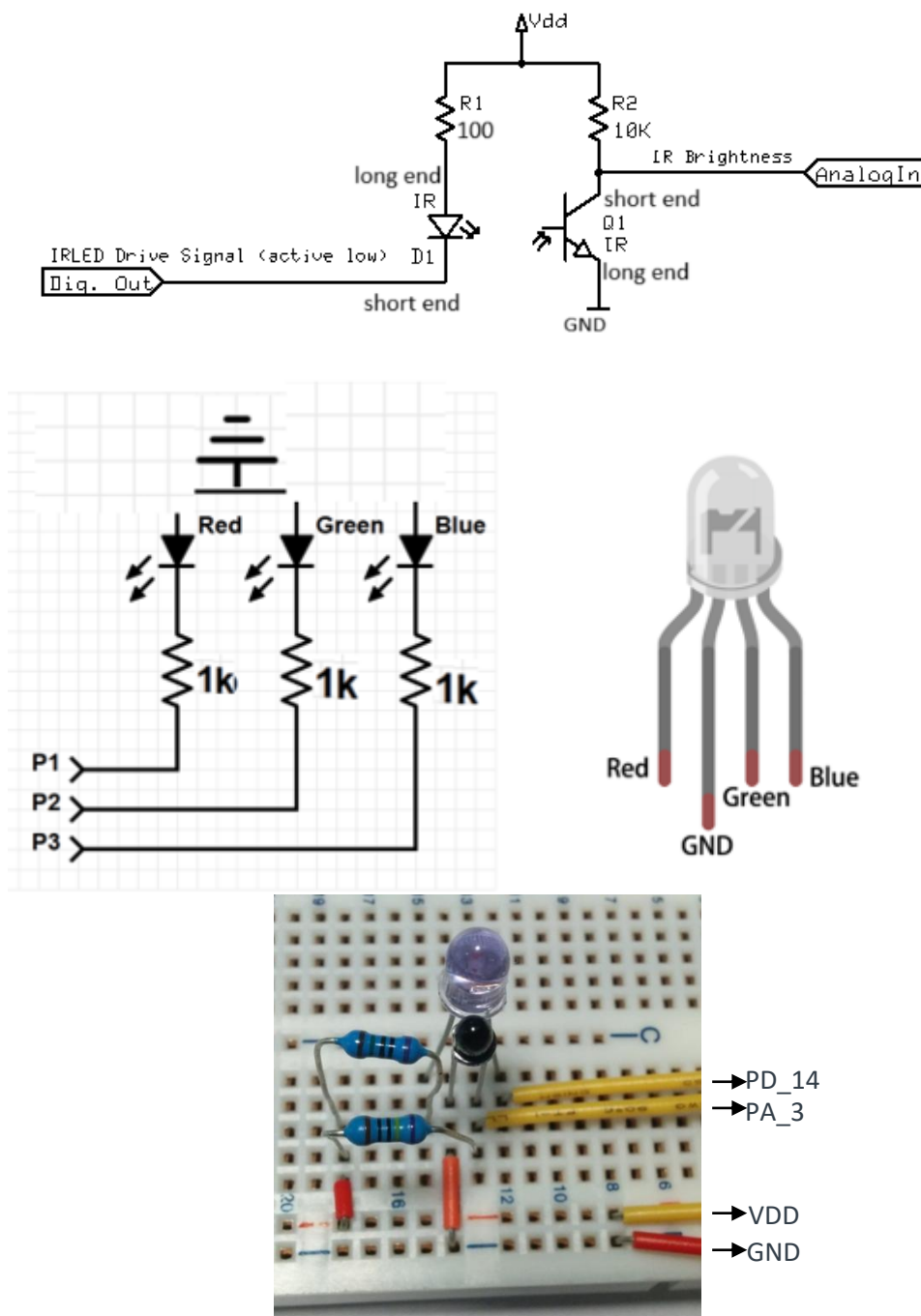


Table 1. Components


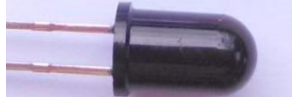
Identifier	Description	Image
D1 (IR emitter)	Note that the flat side of the package marks the cathode (negative terminal, long lead)	
Q1 (IR receiver)	Note that the flat side of the package marks the emitter (negative terminal, short lead)	

Table 2. Signals and connections

Signal name	Description	Direction	MCU
Dig. Out	Drive signal for IR LED D1	Output	PD_14
Analog In	IR brightness	Input	PA_3
Vdd	3.3V power supply	Power	
GND	Ground	Power	
P_LED_R (P1)	Output to Red LED	Output from MCU	PA_5
P_LED_G (P2)	Output to Green LED	Output from MCU	PA_6
P_LED_B (P3)	Output to Blue LED	Output from MCU	PA_7

The 100 ohm resistors should have 3 black bands in the middle. 1k ohm should have 2 black bands and a brown, while 10k should have 2 black bands and a red!

Once you start taking measurements in the lab, try to minimize touching the IR receiver and emitter! The values you obtain can change substantially if the relative positions of the two are changed!

3.2 Software

The control software has the following functions:

- Initialization function: configures GPIO pins and ADC input as needed.
- Delay function: performs delay loop based on function argument.
- Pick LEDs function: Lights RGB LED according to input argument (IR brightness difference).
- Measure IR function: measures difference in brightness caused by lighting LED. In order to reduce noise, average at least twenty measurements before each update of the LED color.
- Main function: Initializes system and then repeatedly calls measure IR, which is passed to the pick LEDs function.

4 Procedure

4.1 Basic Steps

1. Load the project into Keil and build it. Download it to the Nucleo-L552ZE-Q board in debug mode.
2. View the “ir.c” file and add the following variables to the watch window. Enable periodic window updates in the View menu. Remember that only static variables can be monitored live, and that you’ll have to first have trigger breakpoint in “ir.c” (each time you restart the program!) such that these values are initialized before you’ll be able to see any values.
 - reads
 - background_ir
 - differences
 - avg_diff

4.2 Analysis

4.2.1 Ambient Light Analysis

3. Disconnect the IR emitter. Is the photo transistor (IR receiver) reading of IR brightness steady or does it vary? If so, how much and why?
 - a. The reading of IR brightness is steady. We noticed it does not change more than +/- 10. This is because there no difference in brightness it can measure.
4. Reconnect the IR emitter and monitor the IR difference reading. Shield the photo transistor (IR receiver) from IR energy emitted from the side of the LED (aka place a piece of paper between the two). Does this change the difference signal strength? If so, by how much and why?
 - a. The piece of paper made the whole sensor work much less efficiently. The difference value was close to zero while the paper was between the emitter and receiver. This is because the receiver is not able to take an any emitted infrared light.

4.2.2 Time Delay Analysis

5. The sensitivity of the proximity sensor increases as you wait longer to sample the photo transistor’s voltage after changing the IRLED (see Figure 2 above). What is the impact of doubling the default on-time and off-time? Adjust the delays in the ir.c code to accomplish and then test this, comparing the initial results for an object of your choosing at a distance of your choosing to the same object at roughly the same distance after changing the code.
 - a. When we double the delay, we noticed that the time it chose to read the IR level, and the background IR level was longer. The impact of this that it does not update as frequently, and we had to spend more time waiting for the distance values to be the same.

Change these values back to the default before moving on!

4.2.3 Range Analysis

6. What is the maximum distance at which your sensor can reliably detect a sheet of paper? Your hand? Your thumb?
 - a. We noticed that it could detect the sheet of paper around 14cm and it was able to detect my hand at a much shorter distance of 8cm. When we tried to detect our thumb, it was much more difficult because of the size compared to the piece of paper and the hand. We only got a distance of around 4cm before it would read nothing.

4.2.4 Calibration

7. Calibrate your code so that the RGB LED is lit according to object distance, shown below. Calibrate by adjusting the values in the array called threshold, declared in main.c. To ensure consistency, use the same object for all calibration and testing.
 - Green: No object within 20cm
 - Blue: 16-20 cm
 - Yellow: 12-16 cm
 - Red: 8-12 cm
 - Magenta: 4-8 cm
 - White: 0-4 cm

```
const int threshold[] = {65, 49, 32, 24, 10, 0};
const int colors[][3] = {
  { 1, 1, 1 }, // White
  { 1, 0, 1 }, // Magenta
  { 1, 0, 0 }, // Red
  { 1, 1, 0 }, // Yellow
  { 0, 0, 1 }, // Blue
  { 0, 1, 0 }  // Green
};
```