Report of the 1st Project - Sensor Fusion for Proximity Estimation

Diogo Matos, 95778
Instituto Superior Técnico
Santo Tirso, Portugal
diogo.s.matos@tecnico.ulisboa.pt

Daniel Pacheco, 96170
Instituto Superior Técnico
São Brás de Alportel, Portugal
daniel.pacheco@tecnico.ulisboa.pt

José Antunes, 96258
Instituto Superior Técnico
Lisboa, Portugal
jose.m.valerio.antunes@tecnico.ulisboa.pt

***Abstract*—In this project, we developed a device capable of measuring its distance to an object in front of it using an Arduino Nano 33 BLE, an ultrasonic sensor, an IR led, a photo diode and a few electronic components. This device used both ultrasonic waves and IR light to measure its distance to the object, and with the data retrieved from the IMU inside the Arduino, was capable of compensating for any tilting of the device. An extra feature was also added to our device that allowed it to recognize if the object in front of it was bright or dark and display that information using the embedded LEDs.**

***Index Terms*—ultrasonic sensor, IR light, distance, IMU, Arduino.**

I. Introduction

The project developed had the objective of creating a device whose main purpose was to measure distances. This was to be achieved by fusing data from various types of sensors. The Arduino Nano 33 BLE used serves as the main processor, executing the program provided and retrieving all the sensor's information. The sensors provided were an ultrasonic sensor, an IR led together with a photo diode and the Arduino's embedded IMU. In this report, we will denominate the combined use of the IR led and the photo diode as the IR sensor. The ultrasonic sensor and the IR sensor serve as proximity sensors. As the name suggests, the ultrasonic sensor uses sound as a way of measuring distances, whereas the IR sensor uses infrared light. So, each one of the proximity sensors has a different physical principle behind its working, and therefore different strong and weak points.

The ultrasonic sensor works by emitting a pulse and checking how long its echo takes to reach the sensor. By analyzing this time, we can calculate how far the object is since the sound's air travel speed is a known constant. This sensor does not depend on environmental factors, such as light intensity or particles in the air. However, if the surface that the sensor is emitting to is not flat, the reflection of the sound waves (echo) may not return to the sensor, and the distance cannot be calculated. Also, if the material is soft, some of the sound waves will be absorbed, making the echo weaker. There is also a dead zone in the ultrasonic sensor. This zone is situated directly in front of the sensor and is caused by the continued vibration of the transducer, a component of the ultrasonic sensor. This vibration must dissipate before the transducer listens to another echo; if it does not, the distance will not be calculated accurately. Taking this into account, the ultrasonic sensor won't perform well when trying to detect edges or at very close range.

The other proximity sensor used was the IR sensor. As mentioned before, it consists of an IR led and a photo diode. The IR led has the solo task of emitting infrared light while the photo diode transforms light into current. The diode is especially sensitive to infrared light. This sensor uses light reflection as its basis. The light emitted by the IR led will be transformed into current by the photo diode after being reflected on an object. The closer the object is, the more current will be generated. This current passes through a circuit which will transform it into a voltage, which will be further amplified, filtered and analyzed by the Arduino. However, the relation of current/distance is not consistent with all objects since the light reflected depends on the object's colour. There is also the "noise" that the photo diode captures, such as the sunlight coming through a window. Due to this reason, the distance measured by the IR sensor is only accurate when at close range.

The last sensor is the Arduino's embedded Inertial Measurement Unit (IMU). Its purpose is not to measure distances but to measure accelerations on the x, y and z-axis. These accelerations are necessary, more specifically, the acceleration in the y and z coordinates, to calculate the Arduino's tilt and the correction that needs to be made to the distance obtained by the proximity sensors.

II. Theory

The Arduino 33 BLE has eight analogue and fourteen digital pins that allow it to communicate with external devices. We will use digital pins to send and retrieve signals from the ultrasonic sensor and an analogue pin to retrieve the signal from the photo diode. It also has a 3.3V output pin that will be used to power the IR led and a 5V output pin that will power both the ultrasonic sensor and the OPAmp. In the analogue pins, we made use of a 10-bit Analogue to Digital Converter (ADC), as we felt that the goal at hand would not have benefited from the added precision of the 12-bit ADC.

The ultrasonic sensor, when a pulse of at least 10 µs is applied to its Trigger pin, emits a burst of sonic pulses with a known signature that allows the system to differentiate it from interfering signals. While the pulse travels through the air, the

Echo pin goes high. Once the signal is received, the Echo pin goes back to low, timing out after 38 ms. This allows us to measure the distance between the sensor and an object in front of it through the following equation, where v_{us} represents the velocity of sound through the air:

$$distance = v_{us} \times time/2 \tag{1}$$

Since we are using the time until the sensor detects the echo of the burst signal, then we have to consider the time of travel to the object and back, and as such, we have to divide the time interval by two to get an accurate measurement. Additionally, the speed of sound considered in our implementation was $340m/s$.

The IR led emits IR light once powered on. To power it, we must use a voltage power of 1.6V. In order to do so, we connected one end of a 43 Ω resistor to the 3.3V output pin of the Arduino and its other end to the input pin of the LED.

The photo diode is responsible for converting radiance into current. It is especially sensitive to IR light, which allows it to effectively detect light emitted by the IR led. However, we cannot measure current in the Arduino device, and the values of the current output of the diode are very low (around a few μA), which means we will have to transform this signal into something the Arduino can detect and read.

In order to do so, we use a simple transimpedance amplifier circuit, which can be seen in the attachment 1:

The resistor Rf will have to be in the order of MΩs to convert the diode current into a voltage high enough to occupy the full extent of the Arduino's ADC range, which goes up to 3.3V, but low enough not to damage the Arduino since it shouldn't receive voltages higher than 3.3V in its analogue pins.

The signal at the end of the transimpedance circuit is then directed to a low-pass filter. This way, we ensure the signal is stable enough to be properly read by the ADC at the analogue pin of the Arduino.

The Arduino Nano 33 BLE, as mentioned, has an embedded Inertial Measurement Unit (IMU), which includes different sensors, such as a 3-dimensional accelerometer, a 3-dimensional gyroscope and a 3-dimensional magnetometer. With the use of the Wire library and the I2C protocol, we obtained the values for the acceleration in the three dimensions in order to be able to determine the inclination of the sensor. Since both sensors are aligned with the y direction in the Arduino, we can obtain the angle of the sensor with the following:

$$\alpha = atan(a_y/a_z). \tag{2}$$

With the angle we can then obtain the corrected distance with the angle:

$$distance_{corrected} = distance \times cos(\alpha). \tag{3}$$

The Arduino Nano 33 BLE also has 3 LEDs that can be turned on or off using the digital pins 22, 23 and 24 for the red, green and blue LEDs, respectively. These will be used to display the brightness of the object in front of the device. For a bright object, we decided to display a white colour, which is achieved by activating all three LEDs. For a dark object, we only turn on the blue LED. For an object with an intermediate brightness, we display a yellow colour by turning on both the green and red LEDs.

III. Implementation

A. Implementing the Ultrasound Sensor

In order to implement the readings of the ultrasound sensor without active wait to read the Echo signal, which we are unable to use due to the need to also run other measures at the same time (IR measures and IMU measures), we implemented an asynchronous method with the use of interruptions.

As such, we attached an interrupt to the Echo pin that runs our target function on signal change. Afterwards, when we activate the trigger in the ultrasound, which sets the Echo pin to High, it runs, storing the start time, and when the Echo pin finally changes to Low, it gets the end time, and with the difference between both, calculates the duration of the echo to be detected, which can then be used to calculate the distance. By attaching the interruption, it allows for this target function to be executed whenever the Echo pin changes, and as such, the Arduino can still execute other functions in the meanwhile.

We also add a 1cm offset to the distance measured by this sensor, since it experimentally improved its accuracy.

B. Creating the circuit to obtain IR measurements

As stated in the Theory section, we used a transimpedance circuit and a low pass filter to transform the signal created by the photo diode into a signal that can be used by the Arduino. In the transimpedance circuit, we used a $1.3M\Omega$ resistor. This value was chosen because it belonged to the range of values we decided were appropriate for this circuit, as was explained in the Theory section, and experimentally provided excellent results since it used the full range of the analogue input ADC, which we can observe in attachment I.

To create the low pass filter, we used a $120K\Omega$ resistor and an 82nF capacitor. Using the formula in equation 4, we can calculate a cutoff frequency of 16.2 Hz for this filter, which is much lower than the frequency of the ADC and, therefore, suited for this application.

$$f_c = 1/(2 \times \pi \times R \times C) \tag{4}$$

C. Usage of the IR measurements

In order to properly use the IR led and photo diode to measure the distance to an object, we had to create experimental models to correlate these variables. We used different materials and measured the analogue input for distances between 11cm and 2cm, with points spaced by 1cm. The results are shown in table I located in the attachments.

With this data, we were able to calculate, using Excel, expressions that, for each material, would approximate its distance to the device given the analogue input. Plots of the data and the corresponding expressions are represented in 2.

With this data, we were able to create five models that gave us a distance prediction based on the analogue input. To use these models, we created an algorithm that uses the following assumptions: The ultrasonic measurements are very accurate between 10cm and 5cm and the perfect model for any object can be written as a linear combination of the two closest models we have access to.

Using these assumptions, we created an algorithm that calibrates the IR measurements. This algorithm waits until the ultrasonic distance readings are between 10cm and 5cm to calibrate the IR measurements. Once that range of distances is achieved, it reads the analogue input, calculates the output of each one of the five models and selects the two models with the closest outcome to the distance measured by the ultrasonic sensor. The weights of the other three models are set to 0, while the weights of these two models are calculated using the expression below, where "1" represents the model with the best approximation, "2" represents the model with the second best, "w" represents the weight attributed to a model, "d" the distance calculated by it and "distance" the distance obtained using the ultrasonic sensor:

$$w_1 = |(distance - d_2)/(d_1 - d_2)| \tag{5}$$

$$w_2 = 1 - w_1 \tag{6}$$

Once we calibrate the IR measurements, we can use them to calculate the distance with the following expression, where w_i represents model i's weight and d_i represents model i's calculated distance:

$$distance_{IR} = \sum_{i=0}^{4} w_i \times d_i \tag{7}$$

We only re-calibrate the sensor if, right after calibration, the difference between the distance obtained by ultrasonic and IR measurements is greater than 0.7cm or if the difference between these distances is greater than 1cm for the calibration distance (between 10cm and 5cm).

To obtain more consistent and accurate distance values, instead of calculating a distance for every measure, we take the average value of 100 measures to perform this operation.

D. Analysing the brightness of the object

With the previous algorithm, we are also capable of estimating the brightness of the object in front of the device. Since each model was made for an object of different brightness, and the main characteristic that makes an object approximate one model is its brightness, by choosing the model that more closely approximates the object's behaviour, we are also choosing the brightness of the object facing the device. To display this information, as stated before, we used the LEDs embedded in the Arduino device. To turn these lights on or off, we have to perform digitalWrite() on the corresponding pins. If we write HIGH, the led turns off. If we write LOW, it turns on.

At the beginning of the program, we turn off all LEDs by writing HIGH on all corresponding pins. Then, when we perform the calibration, once we figure out which model calculated the most accurate distance, we perform digitalWrite() on all LED pins in order to display the correct colour. These colours will be displayed until a new calibration is done.

E. Communication with the IMU

To communicate with the IMU, we utilized the Wire library from Arduino and the I2C protocol.

For this, we considered the Arduino as the master device and the LSM9DS1 IMU sensor as the slave, and we started by initializing the Wire1 object since we are running the sensor embedded in the Arduino Nano. We then proceeded to reset the sensor, followed by the configuration of the range of $\pm$ 2g since we are only using this acceleration to determine an angle, and in normal circumstances, the absolute of the acceleration will not be higher than 1g.

In order to do so, we had to write into specific control registers identified in the datasheet of the sensor [1]. As such, the protocol demanded that we write into a buffer later sent to the slave device 2 bytes, the first one identifying the register to be written to and then the value that was to be written.

Then, we can finally start reading the acceleration values. This is done by first selecting the register from which we want to read, and we can do this by writing into the slave device a byte with the address of the data register we want. Following that, we can request from the slave address the number of bytes we want, which in our case is 6, since each of the dimensions for the acceleration is 16-bit long or two bytes, and that allows us to read six sequential bytes to read all of the values that we want, from the bytes we requested. We then have to convert the binary values we obtained into a signed integer of 16 bits to obtain the values of the acceleration in the desired range.

F. Fusion of the Sensors

Our distance sensor was composed of a combination of an ultrasound sensor and an IR sensor, and as such, we had to establish a mechanism to fuse the results from both sensors in order to obtain the desired final distance.

To do so, we established some conditions which allowed us to decide which measurements to choose from. Above 10cm of distance, because only the ultrasonic sensor is capable of providing accurate results, since the ambient light makes the IR light reflection almost undetectable in these conditions, the final distance would be the one obtained from the ultrasound sensor. Below that, as mentioned, and between 10cm and 5cm, the distance from the ultrasound would be used to calibrate the IR sensor, and from the moment it was considered calibrated, below 10cm, the value used would be from the latter. If the distance was below 10cm, but the IR sensor was not yet calibrated, then the ultrasound distance would be used. Both of these distances were calibrated for the inclination of the sensors.

Additionally, as will be mentioned below, we observed a high error when the inclination was high, and we were

measuring from the IR sensor since it was highly directional, and any inclination severely impacted the IR light detected as it would be reflected in another direction. As such, we also opted to use the ultrasound distance as the final in the following situations: if the inclination was higher than $\alpha = 0.4$rad; or for distances between 10cm and 5cm and the inclination was higher than $\alpha = 0.1$rad.

We also discarded distance measures when the analogue input has too high to be properly measured, which we decided would happen when it was higher than 1020. Since this only happens when the object is very close to the device, using the ultrasonic measurements would also lead to inaccurate values, meaning there is no way to properly obtain the distance in these conditions.

IV. Results

Looking at the results which can be observed in the graphs presented in 3,4 and 5 for differently coloured objects, we can understand that the sensors are working as desired withing reasonable margin. The values obtained by the fused sensor are very close to the real values for the short distances evaluated (20cm and below), with a maximum error of around ± 0.5cm.

Taking into account the nature of the sensors in use, we find these results to be very acceptable when considering their limitations and flaws. This demonstrates that the algorithm used to calibrate the IR sensor was adequate and that the models used represented our reality accurately, even when the colour of the object was not included in the set used to create the models.

A. Sources of Errors

We can, however, highlight various sources for errors that we encountered.

Firstly, the measurements are very unstable, varying a lot for the same object at the same distance and sometimes showing a bigger error margin than what can be observed in the graphs, as these were in ideal conditions and had good calibration.

Additionally, the calibration process was not always perfectly accurate, since it was based on the distance measured by the ultrasonic sensor. This would result in an accumulated error that affected the final error obtained. This could be mended by performing the calibration often until the desired value was achieved. However, this can only be done if we have a sensor accurate enough to be considered as ground truth, which is not the case.

This calibration process was very dependent on the testing environment conditions, as it was only relevant for the IR sensor, which was also affected by the intensity of the ambient lighting. As such, the brighter the environment, the higher the base value would be for the tension detected by the Arduino, and similarly, the more intense the reflected radiation would be, further altering the distance values produced, as the models were devised in very specific conditions in the laboratory. A possible solution for such a problem would be the more extensive testing and devising of more models of more colours and different base lighting conditions to make the calibration and, therefore, the measured IR distance more robust. However, as long as the object's real model was within the area covered by all the existing models, we found the calibration to be robust enough to maintain errors below 0.5cm.

Finally, another significant issue that affected our measurements was one mentioned above: the angle that the test object made with the sensors. This issue mostly affected the IR measurements, as the sensor used was directional and easily affected by a small angle with the object, meaning that the radiation reflected detected on the photo diode would be less than is in reality, and therefore, considerably affecting the resulting distance.

V. Conclusion

To sum up, in this laboratory, we created a device that, using an Arduino Nano 33 BLE, its IMU, an ultrasonic sensor and an IR sensor, was capable of measuring its horizontal distance to an object. When that object was oriented in such a way that the reflection of the ultrasonic bursts and IR light was directed to the device, this distance was very accurate. However, when this was not the case, the distance predictions could be very inaccurate, especially with the IR sensor. Apart from this limitation, which is not possible to overcome with the sensors we used, the device provided great results, measuring its distance to a multitude of objects with an error inferior to 0.5cm and correctly compensating for any tilt suffered.

Appendix

Attachments

Attachment 1

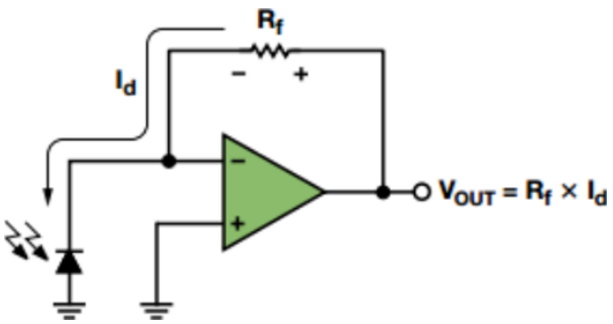

Fig. 1: Transimpedance Amplifier Circuit

Attachment 2

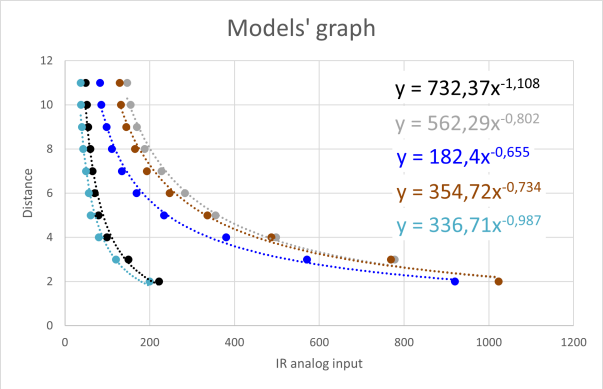

Fig. 2: Graphics and expressions that model the relation between the distance and intensity

Attachment 3

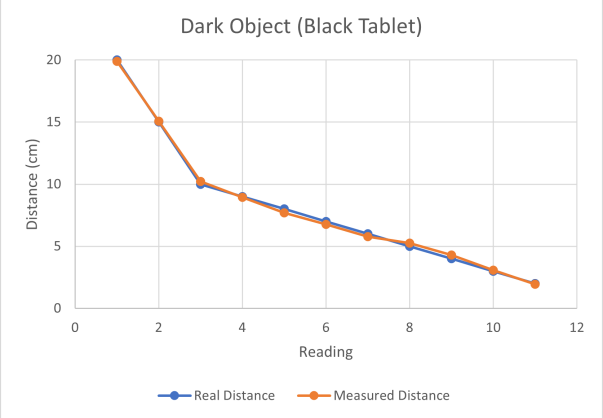

Fig. 3: Comparison between measured and real distance with a dark tablet.

Attachment 4

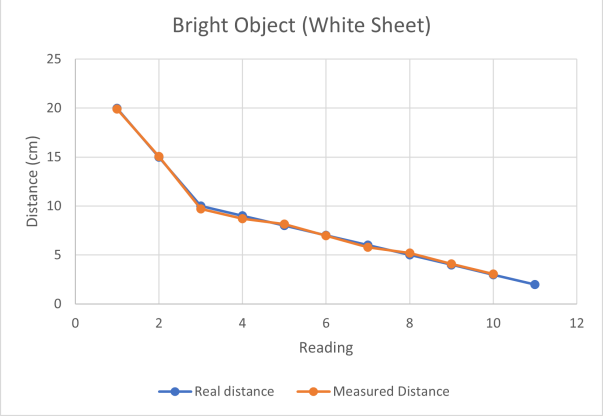

Fig. 4: Comparison between measured and real distance with a white sheet of paper.

Attachment 5

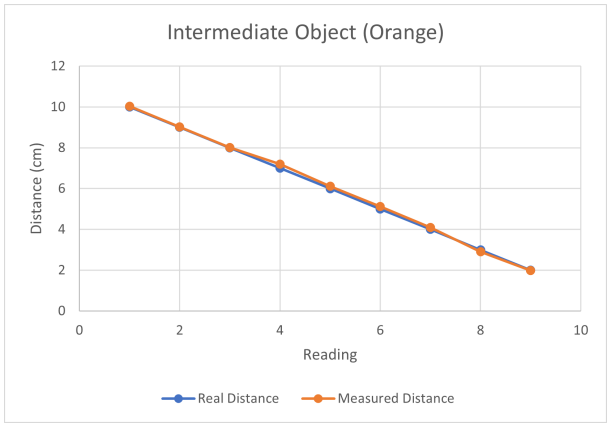

Fig. 5: Comparison between measured and real distance with an orange book cover.

Attachment 6

Distance [cm]	IR analog input				
	Object 0	Object 1	Object 2	Object 3	Object 4
11	49	147	83	129	37
10	52	155	86	132	38
9	55	170	98	145	40
8	60	188	111	165	43
7	65	228	134	193	50
6	70	283	169	247	56
5	79	355	234	336	61
4	99	498	380	487	80
3	150	778	571	769	120
2	222	1023	920	1023	200

TABLE I: Relation between distance and the input received by the Arduino.

REFERENCES

[1] ST, "LSM9DS1 9-axis iNEMO inertial module (IMU)," [Online]. Available: https://www.st.com/en/mems-and-sensors/lsm9ds1.html.

[2] Arduino, "Arduino General https://www.arduino.cc/.

[3] Nordic, "nRF52840 Product Specifications," Nordic, [Online]. Available:https://content.arduino.cc/assets/Nano_BLE_MCU-nRF52840_PS_v1.1.pdf.

[4] Arduino, "Arduino Nano 33 BLE product page,"[Online]. Available: https://docs.arduino.cc/hardware/nano-33-ble.

[5] A. Electronics, "Description of the HC-SR04 Ultrasonic Sensor," [Online]. Available: https://ampere-electronics.com/p/hc-sr04-ultrasonic-sensor-module-3/.

[6] A. Electronics, "Description of the HC-SR04 Ultrasonic Sensor," [Online]. Available: https://ampere-electronics.com/p/hc-sr04-ultrasonic-sensor-module-3/.

[7] Vishay, "High Speed Infrared Emitting Diode, 890 nm," [Online]. Available: https://www.vishay.com/docs/81040/tssf4500.pdf.

[8] Vishay, "Silicon PIN Photodiode TEFD4300," [Online]. Available: https://www.vishay.com/docs/83471/tefd4300.pdf.

[9] T. Instruments, "OPAx340 Single-Supply, Rail-to-Rail Operational Amplifiers," Texas Instruments, [Online]. Available: https://www.ti.com/lit/ds/symlink/opa2340.pdf.

[10] L. Orozco, "Optimizing Precision Photodiode Sensor Circuit Design," Analog Devices, [Online]. Available: https://www.analog.com/en/technical-articles/optimizing-precision-photodiode-sensor-circuit-design.html.

[11] T. Mohammad, "Using Ultrasonic and Infrared Sensors for Distance Measurement.," in World Academy of Science, Engineering and Technology, Hong Kong, 2009.

[12] Arduino, "Arduino Language Documentation," [Online]. Available: https://www.arduino.cc/reference/en/.
[13] GETTING STARTED WITH THE ARDUINO – CONTROLLING THE LED (PART 1): https://www.circuitbasics.com/arduino-basics-controlling-led/.

Daniel Pacheco graduated in Eletrical and Computer Engineering from Instituto Superior Técnico in Lisbon, Portugal, in 2022. He is currently pursuing an M. S. in Electrical and Computer Engineering, majoring in Computer Systems with as minor in Networks and Communication Systems. Currently works in JEEC, being the leader of the Web Dev team since 2022, and part of the team since 2021. Also worked for the After School project between December 2021 and May 2022, and took an internship in Aptoide during the summer of 2022. His research interests are Parallel and Heterogeneous Computing, High-performance Computing and Software Optimization.

Diogo Matos received his B. S. in Aerospace Engineering from Instituto Superior Técnico in Lisbon, Portugal, in 2022. He is currently pursuing an M. S. in Electrical and Computer Engineering, majoring in Control, Robotics and Artificial Intelligence, with a minor in Computer Systems. Currently is part of the Board of the European Youth Parliament Portugal and has taken an internship as a software developer at MorphisTech in 2021. His research interests are Heterogeneous Systems, Artificial Intelligence Accelerators, and RISC-V.

José Antunes received his B. S. in Electrical and Computer Engineering from Instituto Superior Técnico in Lisbon, Portugal, in 2022. He is currently pursuing an M. S. in Electrical and Computer Engineering, majoring in Computer Systems with a minor in Control, Robotics and Artificial Intelligence. Currently works in JEEC, being a part of the Web Dev team since September 2021. His research interests are Quantum Computing, GPUs and Machine Learning.