TÉCNICO LISBOA

Technologies of Computing Systems
MEEC

1st Project

Sensor Fusion for Proximity Estimation

1st Semester, Second Quarter 2022/23

Leonel Sousa, Diogo Caetano, Ruben Afonso

November 2022

1 Introduction

This project includes the assembly, testing and calibration of a proximity sensing system. The base of the design is an Arduino [1] electronic platform, equipped with an nRF52840 Microcontroler bearing an Arm Cortex-M4 32-bit Processor [2], a 12-bit ADC, and an internal inertial measurement unit, among other things. Additional information about the specific Arduino platform to be used in this project (i.e. Nano 33 BLE) can be found in the products page [3].

This project uses two different external proximity sensors, based on distinct physical principles. One is based on ultrasonic time of flight and the other in infrared refraction intensity. Ultrasonic sensors are mostly independent from environmental factors like, light intensity, dust, smoke, etc. However Ultrasonic sensors are not ideal for edge detection or close distance measurements. Infrared sensors operate on the principle of reflected light intensity. An infrared emitter produces the output that is consequently reflected by objects in the vicinity, and there is a correlation between the light intensity and the objects proximity. Infrared sensors are usually more accurate for close distance measurements. However, infrared sensors are very susceptible to reflecting object's color and external light sources that contain energy in the infrared wavelength (e.g., sunlight), as they produce strong interference.

The inertial measurement unit (IMU) among other things can be used to measure acceleration, and on the earth's surface it can be used to determine the devices angle regarding the earths gravitational force. In this project the IMU should be used to compensate for the systems tilt that induces errors in the proximity measurements

2 Material and devices

For this work the following materials and components are necessary:

i. Breadboard
ii. Arduino Nano 33 BLE
iii. USB Cable
iv. Ultrasonic Sensor – HC- SR04 [4]
v. IR led (TSUS5200 - 950nm or TSHF6410 – 890 nm) [5]
vi. Photo diode (BPV10 380nm to 1100nm or TEFD4300 - 350nm to 1120nm) [6]
vii. Operational Amplifier (OPA340) [7]
viii. Resistors
ix. Capacitors

3 Goals

This project targets the following main objectives.

i. Implement the ultrasonic sensor system, including the interface to the Arduino platform and calibrate the system to obtain an accurate distance measurement. With that purpose, it must:
   a) produce evenly spaced impulses.
   b) take advantage of interrupts to produce a new pulse as soon as the eco signal is received.

ii. Implement the infrared sensor system, applying biasing and amplification electronics [8] (discrete components). Convert sensor response to distance and calibrate the system to obtain an accurate distance measurement.

iii. Use both sensors in collaboration.
   a) Choose the optimal sensor for the estimated distance.
   b) Use weighted average for the distance estimation e.g., Ultrasonic output has more weight for longer distances (e.g., >15 cm) and the infrared for shorter distances.

   The goal is to achieve better precision in the full distance range. Determine the optimal range for each sensor.

iv. Compensation Mechanisms:
   a) Accelerometer (IMU) is used to compensate for sensor tilting and for calculating the real distance to obstacle relative to the systems geometric center.
   b) Ultrasonic based measures are used to compensate infrared response to different refraction rates of objects with different colors.

4 Sensors

As referred before, in this project two sensor with different physical principles are used for estimating the object proximity: the HC-SR04 ultrasonic based sensor [4], the Infrared sensor (IR emitter [5] and photodiode [6]), and the LSM9DS1 [9], an inertial module available at the Arduino 33 BLE board which should be used to detect the system tilting.

4.1 Ultrasonic sensor

The HC-SR04 [4] employs ultrasonic echo-based distance estimation. The proximity measurement starts when a pulse of at least 10 µs is applied to the Trigger pin of the module. The sensor will emit a burst of sonic pulses with a known signature that allows

the system to differentiate from interfering signals. The pulses travel through the air (Figure 1). During this time the Echo pin goes High. If there is no reflection the Echo pin will timeout after 38ms. If the emitted pulses are reflected, the echo pin goes low as soon as the signal is received. This produces a pulse duration related to the distance of the reflecting obstacle, which can be used to compute the said distance.

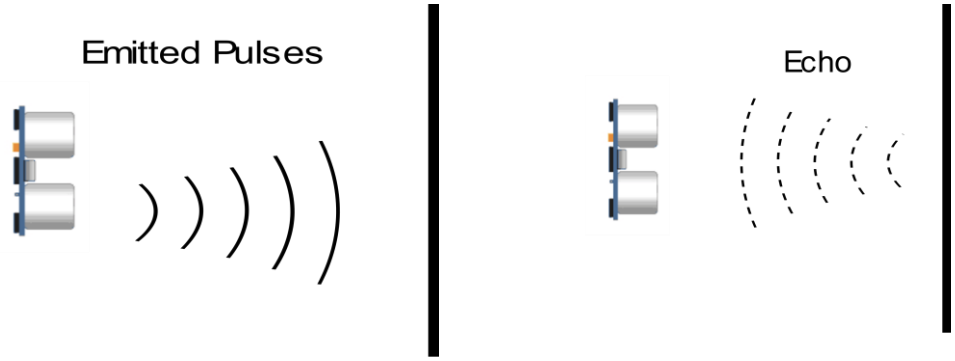

Figure 1: Ultrasonic signal emission and subsequent eco.

4.2 Infrared sensor

The infrared sensor uses infrared light reflection as a means of proximity estimation. This sensor is based on an emitter LED [5] and a receiving photodiode [6]. As with the ultrasonic sensor, the emitted light is only reflected in the presence of an obstacle. However, in the case of light, computing the delay between emitted and reflected photons would be demanding. Thus, in this case the distance is correlated with the amount of light that is reflected, and consequently excites the photodiode (Figure 2).

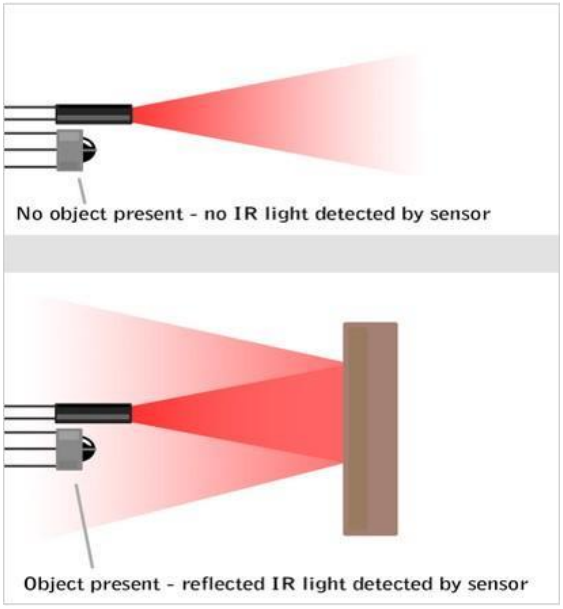

Figure 2: Infrared sensor refraction schematic from.

4.3 Inertial Module

The LSM9DS1 module in the Arduino platform is a 9-axis inertial sensor [9] . It can sense 3 acceleration channels, 3 angular rate channels and 3 magnetic channels. To access the sensor tilt, required in this project, only some of those components are necessary. However, more complex implementations are considered also for the final grade.

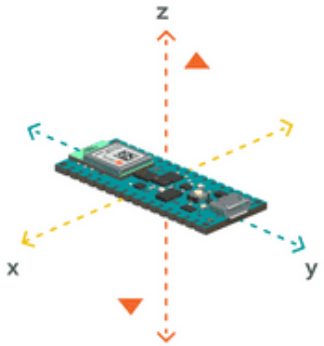

Figure 3: Arduino Nano 33 BLE accelerometer coordinate system.

5 Sensor Fusion

Sensor fusion can be used to improve information retrieval from available data. In this project it can be used to increase precision, compensate for tilt angle, and reflective surface color.

5.1 Precision

Sensors have different precisions for different distances. One of the goals of this project is to create a unified sensor that produces the highest precision for the full range of tested distances (e.g., 0-100 cm). You may use the suggested techniques or derive new approaches.

5.2 Angle Compensation

By observing Figure 4, sensor tilting between the movement plane can introduce significant errors in the eventual traveling distance measure to the actual obstacle. Thus, angle compensation should be introduced, which can be performed by estimating the inclination or tilting angle using the Arduino's accelerometer. The axis orientation of the Arduino board can be seen in Figure 3.

By using the sensor's accelerometer, it is not possible to differentiate between sensor tilt and inclination in the movement plan. Although this will not be contemplated in this project if implemented in an actual autonomous vehicle what other information would be necessary to make this differentiation?

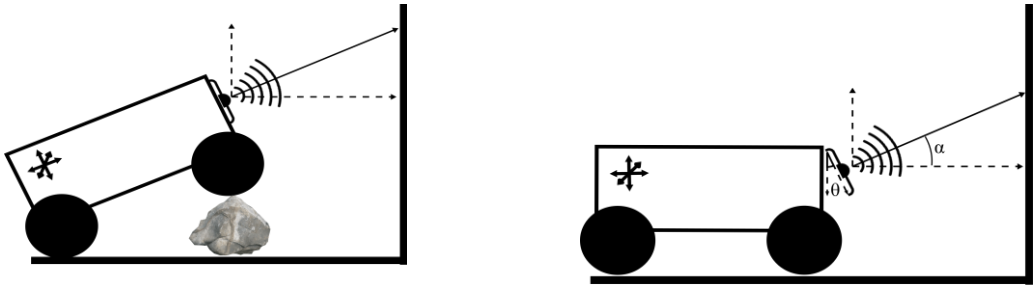

Figure 4: Proximity measurement error sources due to terrain inclination or sensor tilt.

5.3 Color Compensation

Contrary to ultrasonic sensors, infrared sensors are highly dependent on the surface color of the reflecting obstacle. Thus, it is expected that for different colors the sensor will produce different output results. An example of that change can be seen in Figure 5 from [10]. It is clear that except for a very absorbent material color (black) the shape of the curve is similar and there is only an offset. This offset should also be compensated.

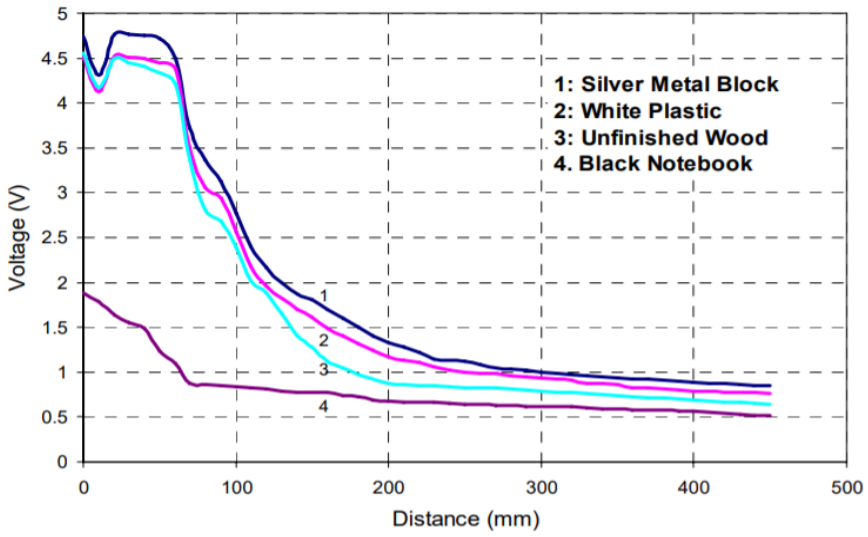

Figure 5: Example of different outputs for of photosensor depending on the reflective surface from *[10].*

6 Implementation details and available libraries

For the implementation of the measurement system in this project, have a set of libraries is available for the Arduino platform.

6.1 Electrical connections

The electronic circuitry developed must be implemented in a single breadboard being powered exclusively by Arduino. The Arduino module must be placed in the breadboard to allow connections with the peripherals, as depicted in Figure 6. There are two voltages available at the Arduino as long as it plug via USB; +5 V voltage are available on the VIN pin (this pin is named VIN because it also accepts external

power in case USB is not connected) and +3.3 V is available on the +3V3 pin. Note that +5V pin is not enabled by default.

CAUTION: The Arduino module core operates with a supply voltage of 3.3V. All input/output pins only support 3.3V. The ultrasound module needs +5V VCC so it is necessary to make a level shift from the ECHO output.

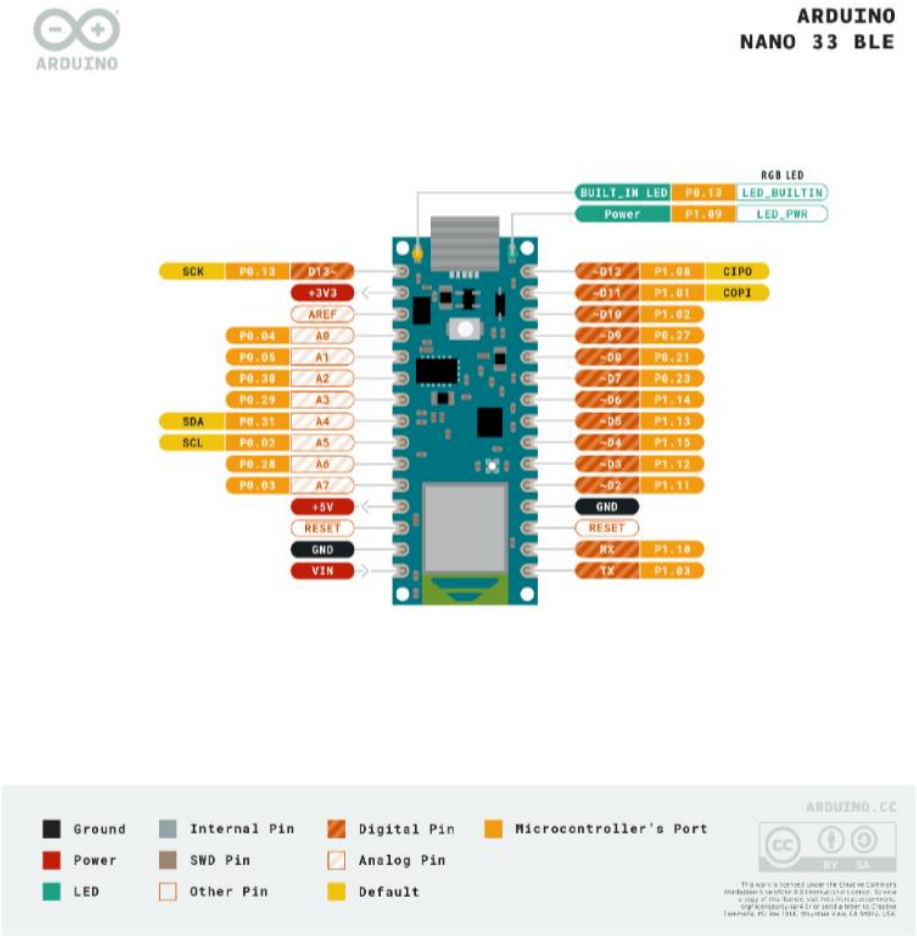

Figure 6: Arduino Nano 33 BLE pinout.:

6.2 Timers

All timed actions must make use of the timer peripheral to set the trigger pulse, measure the duration of the received pulse, and trigger periodic acquisition of the values provided by the ADCs. To use the already developed abstraction layers related to these peripherals, it is possible to include the "timer.h" library (included in the Arduino IDE).

However, if such functions are used, the internal mechanism of each must be explained in the report (e.g., what is the prescaler value, how counts are converted to time, etc.)

6.3 I2C

Arduino has a library called "Wire.h" that assists the handling of serial communications. The I2C communication with the accelerometer must use this library,

by implementing custom read and write functions. The internal mechanism of each function as well as the I2C registers interaction must also be explained in the report.

6.4 Processor Interrupts

Arduino has an abstraction layer that assists to handle of interrupts. External interruptions handler can be set to trigger a function using the function *attachInterrupt()*.

Please note that the interrupt handlers run at priority level APP_LOW. The executed code must be reduced without applying complex operations. If other peripherals are accessed must use nRF52840 HAL functions or inline assembly.

6.5 Libraries

In addition to the libraries referred in the previous subsections, it is still allowed to use serial communication functions to send/receive information to/from the computer. Communication must be started in the setup function after setting the pins using "*Serial.begin(<baudrate>)*". Common values for the baudrate are 9600 or 115200.

Any code section that uses open-source code or libraries (internal or imported) will be disregarded and will not be considered towards the final grade. All the interfaces and data acquisition, generation of interrupts, timers and conversions should be made by the authors. Advance functions such as *noTone()*, *pulseIn()*, *pulseInLong()*, *shiftIn()*, *shiftOut()*, *tone()* and Class Stream should not be used.

6.6 Calibration and complex operations

If calibrations are required, use the multi-segment curves approach: each segment is a linear regression and store the parameters in a lookup table.

The use of float and doubles variables should be minimized, favoring the use of variables in integer formats (e.g. uint8_t, int32_t, long) and, whenever possible, calculations in these formats without losing precision.

6.7 Some practical considerations

i. Start by testing the infrared sensor with a white paper sheet as it is a good reflective surface. The IR receiver cannot be used in direct sun light.

ii. It is necessary to program some configurations/registers in the accelerometer before accessing the values.

iii. It is suggested to use Arduino IDE offline version, installed on the computer.

iv. Do not use the D0 and D1 pins (P1.10 and P1.03) since these are reserved for the serial communication with the computer.

v. The code loading is done automatically by a bootloader programmed in the Arduino module. However, the USB communication uses specific peripheral from the microcontroller's core. If the developed application affects the settings of this peripheral, the connection with the computer may be lost. In this case, Arduino will no longer appear as a serial port in the IDE. The loading

of a new code is only possible by forcing the Arduino to stay in bootloader mode, avoiding running the bad code. This can be done by pressing the reset button twice (< 0.5 sec). The built-in led will initiate a fade-in fade-out sequence to signaling that it is in the bootloader mode.

7 References

[1] Arduino, "Arduino General Website," Arduino, [Online]. Available: https://www.arduino.cc/.

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

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

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

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

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

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

[8] 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.

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

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