RADAR Project Documentation

Embedded Systems Project

Project Title
Mobile radar
A portable Radio Detection And Ranging system

Supervisor
Dr. Brice EKOBO AKOA

Members list

No.	Name
1	DIKOS Johann-Emmanuel
2	EFFOUA Junior
3	OLONGO Cynthia
4	TCHAMKO Chabrell

Abstract

This document will go over the steps we took to build a mobile RADAR system, as well as some of the skills we learned during our time working on this project; the objective being that anyone, after reading it, would be able to build their RADAR and even improve upon what has already been made.

Table of contents:

I- Presentation of the radar

1. Overview and history:

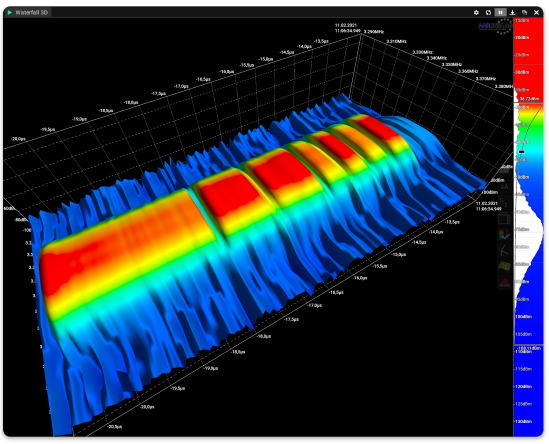
RADAR (an acronym for Radio Detection And Ranging) is a detection system that uses radio waves to determine the distance (ranging), angle, and radial velocity of objects relative to the site. A radar system consists of a transmitter producing electromagnetic waves in the radio or microwave domain, a transmitting antenna, a receiving antenna (a single antenna could receive and transmit), and a processor to determine the properties of the objects. Radio waves (pulsed or continuous) from the transmitter reflect off the objects and return to the receiver, giving information about the objects' locations and speeds. This technology can detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain.

As early as 1886, German physicist Heinrich Hertz showed that radio waves could reflect from solid objects. In 1895, Alexander Popov, a physics instructor at the Imperial Russian Navy school in Kronstadt, developed an apparatus using a coherer tube for detecting distant lightning strikes. The following year, he added a spark-gap transmitter. In 1897, while testing this equipment for communicating between two ships in the Baltic Sea, he took note of an interference beat caused by the passage of a third vessel. In his report, Popov wrote that this phenomenon might be applied to object detection, but he did nothing more with this observation. The German inventor Christian Hülsmeyer was the first to use radio waves to detect "the presence of distant metallic objects". In 1904, he demonstrated the feasibility of detecting a ship in dense fog but not its distance from the transmitter. He obtained a patent for his detection device in April 1904 and later a patent for a related amendment for estimating the distance to the ship. He also obtained a British patent on September 23rd, 1904 for a full radar system, which he called a **telemobiloscope**. It operated on a 50 *cm* wavelength, and the signal was generated by a spark gap. His system already used the classic antenna setup of horn antenna with parabolic reflector and was presented to German military officials in practical tests in Cologne and Rotterdam harbor but was rejected. During the 1930s, efforts to use radio echoes for aircraft detection were initiated independently and almost simultaneously in eight countries that were concerned with the prevailing military situation and that already had practical experience with radio technology. The United States, Great Britain, Germany, France, the Soviet Union, Italy, the Netherlands, and Japan all began experimenting with radar within about two years of one another. They embarked, with varying degrees of motivation and success, on its development for military purposes. Several of these countries had some form of operational radar equipment in military service at the start of World War II.

2. Principles:

a) Radar signal:

A radar system has a transmitter that emits radio waves known as radar signals in predetermined directions. When these signals contact an object, they are usually reflected or scattered in many directions, although some will be absorbed and penetrate the target. Radar signals are reflected especially well by materials of considerable electrical conductivity—such as most metals, seawater, and wet ground. It makes the use of radar altimeters possible in certain cases. The radar signals reflected toward the radar receiver are the desirable ones that make radar detection work. If the object is moving either toward or away from the transmitter, there will be a slight change in the frequency of the radio waves due to the Doppler effect.



Info

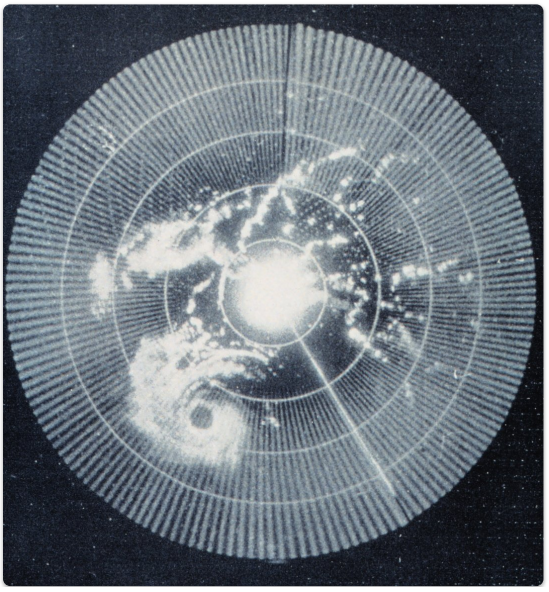
3D Doppler radar spectrum showing a Baker code of 13.

Radar receivers are usually, but not always, in the same location as the transmitter. The reflected radar signals captured by the receiving antenna are usually weak. They can be enhanced using electronic amplifiers. More sophisticated methods of signal processing allow the recovery of useful radar signals.

The weak absorption of radio waves by the medium through which they pass enables radar sets to detect objects at relatively long ranges — ranges at which other electromagnetic wavelengths, such as visible light, infrared light, and ultraviolet light, are too strongly attenuated. Weather phenomena, such as fog, clouds, rain, falling snow, and sleet, that block visible light are usually transparent to radio waves. Some radio frequencies that are absorbed or scattered by water vapor, raindrops, or atmospheric gasses (especially oxygen) are avoided when designing radars, except when their detection is intended.

b) Reflection:

If electromagnetic waves traveling through one material meet another material, having a different dielectric constant or diamagnetic constant from the first, the waves will reflect or scatter from the boundary between the materials. This means that a solid object in air or a vacuum, or a significant change in atomic density between the object and what is surrounding it, will usually scatter radar (radio) waves from its surface. This is particularly true for electrically conductive materials such as metal and carbon fiber, making radar well-suited to the detection of aircraft and ships. Radar-absorbing material, containing resistive and sometimes magnetic substances, is used on military vehicles to reduce radar reflection. This is the radio equivalent of painting something a dark color so that it cannot be seen by the eye at night.



Info

Brightness can indicate reflectivity, as in this 1960 weather radar image (of Hurricane Abby). The radar's frequency, pulse form, polarization, signal processing, and antenna determine what it can observe.

Radar waves scatter in a variety of ways depending on the size (wavelength) of the radio wave and the shape of the target. If the wavelength is much shorter than the target's size, the wave will bounce off in a way similar to the way light is reflected by a mirror. If the wavelength is much longer than the size of the target, the target may not be visible because of poor reflection. Low-frequency radar technology is dependent on resonances for detection, but not identification, of targets. This is described by Rayleigh scattering, an effect that creates Earth's blue sky and red sunsets. When the two length scales are comparable, there may be resonances. Early radars used very long wavelengths that were larger than the targets and thus received a vague signal, whereas many modern systems use shorter wavelengths (a few centimeters or less) that can image objects as small as a loaf of bread.

Short radio waves reflect from curves and corners in a way similar to the glint from a rounded piece of glass. The most reflective targets for short wavelengths have 90° angles between the reflective surfaces. A corner reflector consists of three flat surfaces meeting like the inside corner of a cube. The structure will reflect waves entering its opening directly back to the source. They are commonly used as radar reflectors to make otherwise difficult-to-detect objects easier to detect. Corner reflectors on boats, for example, make them more detectable to avoid a collision or during a rescue. For similar reasons, objects intended to avoid detection will not have inside corners or surfaces and edges perpendicular to likely detection directions, which leads to "odd" looking stealth aircraft. These precautions do not eliminate reflection because of diffraction, especially at longer wavelengths. Half-wavelength long wires or strips of conducting material, such as chaff, are very reflective but do not direct the scattered energy back toward the source. The extent to which an object reflects or scatters radio waves is called its radar cross-section.

c) Radar range equation:

The power P_r returning to the receiving antenna is given by the equation:

$$P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R_t^2 R_r^2}$$

where:

- P_t = transmit power
- G_t = gain of the transmitting antenna
- A_r = effective aperture (area) of receiving antenna, expressed as $\frac{G_r \lambda^2}{4\pi}$ where:
  - λ = transmitted wavelength;
  - G_r = gain of receiving antenna.
- σ = radar cross-section
- F = pattern propagation factor
- R_t = distance from the transmitter power
- R_r = distance from target to receiver.

In the common case where the transmitter and the receiver are at the same location, $R_t = R_r$ and the term $R_t^2 R_r^2$ can be replaced by R^4 where R is the range. This yields:

$$P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R^4}$$

This shows that the received power declines as the fourth power of the range, which means that the received power from distant targets is relatively very small. Additional filtering and pulse integration modify the radar equation slightly for pulse-Doppler radar performance, which can be used to increase detection range and reduce transmit power.

The equation above with $F = 1$ is a simplification for transmission in a vacuum without interference. The propagation factor accounts for the effects of multi-path and shadowing and depends on the details of the environment. In a real-world situation, path loss effects are also considered.

d) Doppler effect:

Frequency shift is caused by motion that changes the number of wavelengths between the reflector and the radar. This can degrade or enhance radar performance, depending on how it affects the detection process. For example, a moving target indication can interact with Doppler to produce signal cancellation at certain radial velocities, which degrades performance.

Sea-based radar systems, semi-active radar homing, active radar homing, weather radar, military aircraft, and radar astronomy rely on the Doppler effect to enhance performance. This produces information about the target velocity during the detection process. This also allows small objects to be detected in an environment containing much larger nearby, slow-moving objects.

Doppler shift depends upon whether the radar configuration is active or passive. Active radar transmits a signal that is reflected; passive radar depends upon the object sending a signal to the receiver.

The Doppler frequency shift for active radar is as follows, where F_D is Doppler frequency, F_t the transmit frequency, V_R is the radial velocity and C is the speed of light:

$$F_D = 2 \times F_T \times (\frac{V_R}{C})$$

Passive radar applies to electronic countermeasures and radio astronomy as follows:

$$F_D = F_T \times (\frac{V_R}{C})$$

Only the radial component of the velocity is relevant. When the reflector is moving at a right angle to the radar beam, it has no relative velocity. Vehicles and weather moving parallel to the radar beam produce the maximum Doppler frequency shift.

When the transmit frequency F_T is pulsed, using a pulse repeat frequency of F_R, the resulting frequency spectrum will contain harmonic frequencies above and below F_T with a distance of F_R. As a result, the Doppler measurement is only non-ambiguous if the Doppler frequency shifts less than half of F_R, called the **Nyquist** frequency since the returned frequency otherwise cannot be distinguished from shifting of a harmonic frequency above or below, thus requiring:

$$|F_D| < \frac{F_R}{2}$$

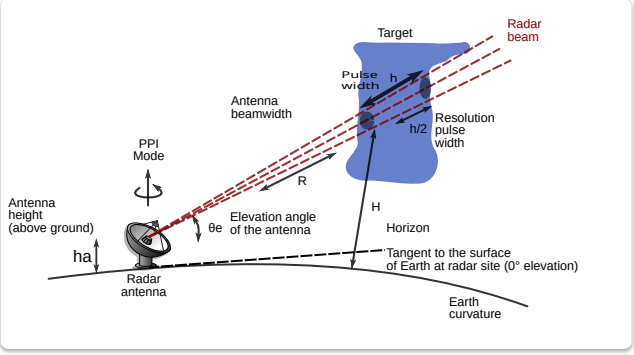
Or when substituting with F_D:

$$|V_R| < \frac{F_R \times \frac{C}{F_T}}{4}$$

As an example, a Doppler weather radar with a pulse rate of 2 kHz and transmit frequency of 1 GHz can reliably measure weather speed up to at most 150 m/s (340 mph), thus cannot reliably determine radial velocity of aircraft moving $1,000$ m/s ($2,200$ mph).

e) Limiting factors:

A radar beam follows a linear path in a vacuum but follows a somewhat curved path in the atmosphere due to variation in the refractive index of air, which is called the radar horizon. Even when the beam is emitted parallel to the ground, the beam rises above the ground as the curvature of the Earth sinks below the horizon. Furthermore, the signal is attenuated by the medium the beam crosses, and the beam disperses.



Info

Echo heights above ground

$$H = (\sqrt{r^2 + (k_e a_e)^2 + 2rk_e a_e \sin(\theta_e)}) - k_e a_e + h_a$$

Where:

- $r =$ distance radar-target
- $k_e = \frac{4}{3}$
- $a_e =$ Earth radius
- $\theta_e =$ elevation angle above the radar horizon
- $h_a =$ height of the feed-horn above ground

The maximum range of conventional radar can be limited by several factors:

- Line of sight, which depends on the height above the ground. Without a direct line of sight, the path of the beam is blocked.
- The maximum non-ambiguous range, which is determined by the pulse repetition frequency. The maximum non-ambiguous range is the distance the pulse can travel to and return from before the next pulse is emitted.
- Radar sensitivity and the power of the return signal as computed in the radar equation. This component includes factors such as the environmental conditions and the size (or radar cross-section) of the target.

3. Signal processing:

a) Distance measurement:

a.1) Transit time:

One way to obtain a distance measurement (ranging) is based on the time-of-flight: transmit a short pulse of radio signal (electromagnetic radiation) and measure the time it takes for the reflection to return. The distance is one-half the round trip time multiplied by the speed of the signal. The factor of one-half comes from the fact that the signal has to travel to the object and back again. Since radio waves travel at the speed of light, accurate distance measurement requires high-speed electronics. In most cases, the receiver does not detect the return while the signal is being transmitted. Through the use of a duplexer, the radar switches between transmitting and receiving at a predetermined rate. A similar effect imposes a maximum range as well. To maximize range, longer

times between pulses should be used, referred to as a pulse repetition time, or its reciprocal, pulse repetition frequency.

Distance may also be measured as a function of time. The radar mile is the time it takes for a radar pulse to travel one nautical mile, reflect off a target, and return to the radar antenna. Since a nautical mile is defined as $1,852\ m$, then dividing this distance by the speed of light ($299,792,458\ m/s$), and then multiplying the result by 2 yields a result of $12.36\ \mu s$ in duration.

b.1) Frequency modulation:

Another form of distance measuring radar is based on frequency modulation. In these systems, the frequency of the transmitted signal is changed over time. Since the signal takes a finite time to travel to and from the target, the received signal is a different frequency than what the transmitter is broadcasting at the time the reflected signal arrives back at the radar. By comparing the frequency of the two signals, the difference can be easily measured. This is easily accomplished with very high accuracy, even for 1940s electronics. A further advantage is that the radar can operate effectively at relatively low frequencies. This was important in the early development of this type when high-frequency signal generation was difficult or expensive.

This technique can be used in continuous wave radar and is often found in aircraft radar altimeters. In these systems, a **carrier** radar signal is frequency modulated predictably, typically varying up and down with a sine wave or saw-tooth pattern at audio frequencies. The signal is then sent out from one antenna and received on another, typically located on the bottom of the aircraft, and the signal can be continuously compared using a simple beat frequency modulator that produces an audio frequency tone from the returned signal and a portion of the transmitted signal. The modulation index riding on the received signal is proportional to the time delay between the radar and the reflector. The frequency shift becomes greater with a greater time delay. The frequency shift is directly proportional to the distance traveled. That distance can be displayed on an instrument, and it may also be available via the transponder. This signal processing is similar to that used in speed-detecting Doppler radar.

Terrestrial radar uses low-power FM signals that cover a larger frequency range. The multiple reflections are analyzed mathematically for pattern changes with multiple passes, creating a computerized synthetic image. Doppler effects are used, which allows slow-moving objects to be detected as well as largely eliminating **noise** from the surfaces of bodies of water.

c.1) Pulse compression:

The two techniques outlined above both have their disadvantages. The pulse timing technique has an inherent trade-off in that the accuracy of the distance measurement is inversely related to the length of the pulse, while the energy, and thus direction range, is directly related. Increasing power for a longer range while maintaining accuracy demands extremely high peak power, with 1960s early warning radars often operating in the tens of megawatts. The continuous wave methods spread this energy out in time and thus require much lower peak power compared to pulse techniques, but require some method of allowing the sent and received signals to operate at the same time, often demanding two separate antennas.

The introduction of new electronics in the 1960s allowed the two techniques to be combined. It starts with a longer pulse that is also frequency modulated. Spreading the broadcast energy out in time means lower peak energies can be used, with modern examples typically on the order of tens of kilowatts. On reception, the signal is sent into a system that delays different frequencies at different times. The resulting output is a much shorter pulse that is suitable for accurate distance measurement, while also compressing the received energy into a much higher energy peak and thus reducing the signal-to-noise ratio. The technique is largely universal on modern large radars.

b) Speed measurement:

Speed is the change in distance to an object with respect to time. Thus, the existing system for measuring distance, combined with a memory capacity to see where the target last was, is enough to measure speed. At one time, the memory consisted of a user making grease pencil marks on the radar screen and then calculating the speed using a slide rule. Modern radar systems perform the equivalent operation faster and more accurately using computers.

If the transmitter's output is coherent (phase synchronized), there is another effect that can be used to make almost instant speed measurements (no memory is required), known as the Doppler effect. Most modern radar systems use this principle in Doppler radar and pulse-Doppler radar systems (weather radar, military radar). The Doppler effect is only able to determine the relative speed of the target along the line of sight from the radar to the target. Any component of target velocity perpendicular to the line of sight cannot be determined by using the Doppler effect alone, but it can be determined by tracking the target's azimuth (The azimuth is the angle formed between a reference direction (in this example north) and a line from the observer to a point of interest projected on the same plane as the reference direction orthogonal to the zenith) over time.

It is possible to make a Doppler radar without any pulsing, known as a continuous-wave radar (CW radar), by sending out a very pure signal of a known frequency. CW radar is ideal for determining the radial component of a target's velocity. CW radar is typically used by traffic enforcement to measure vehicle speed quickly and accurately where the range is not important.

When using a pulsed radar, the variation between the phase of successive returns gives the distance the target has moved between pulses, and thus its speed can be calculated. Other mathematical developments in radar signal processing include time-frequency analysis (Weyl Heisenberg or wavelet), as well as the chirplet transform, which makes use of the change of frequency of returns from moving targets (**chirp**).

a.2) Reduction of interference effects:

Signal processing is employed in radar systems to reduce the radar interference effects. Signal processing techniques include moving target indication, Pulse-Doppler signal processing, moving target detection processors, correlation with secondary surveillance radar targets, space-time adaptive processing, and track-before-detect. Constant false alarm rates and digital terrain model processing are also used in clutter environments.

b.2) Plot and track extraction:

A Track algorithm is a radar performance enhancement strategy. Tracking algorithms provide the ability to predict the future position of multiple moving objects based on the history of the individual positions being reported by sensor systems.

Historical information is accumulated and used to predict future positions for use with air traffic control, threat estimation, combat system doctrine, gun aiming, and missile guidance. Position data is accumulated by radar sensors over a few minutes. There are four common track algorithms:

- Nearest neighbor algorithm
- Probabilistic Data Association
- Multiple Hypothesis Tracking
- Interactive Multiple Model (IMM)

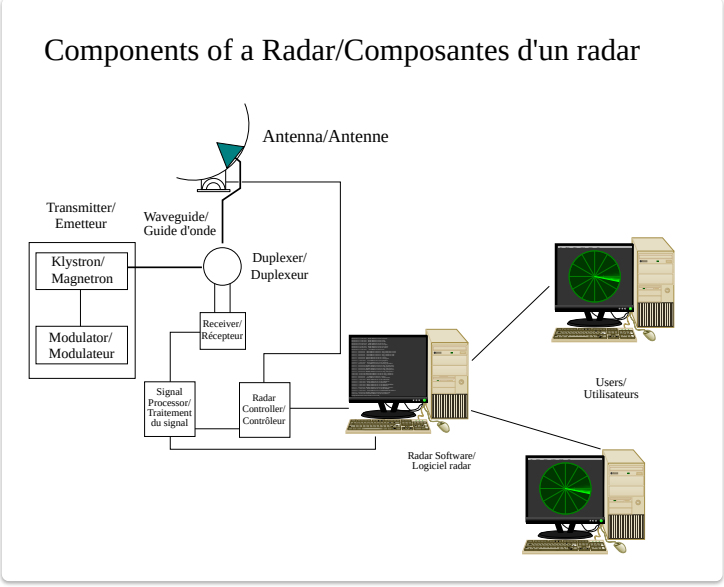
Radar video returns from aircraft can be subjected to a plot extraction process whereby spurious and interfering signals are discarded. A sequence of target returns can be monitored through a device known as a plot extractor.

The non-relevant real-time returns can be removed from the displayed information and a single plot displayed. In some radar systems, or in the command and control system to which the radar is connected, a radar tracker is used to associate the sequence of plots belonging to individual targets and estimate the targets' headings and speeds.

4. Engineering:

A radar's components are:

- A transmitter that generates the radio signal with an oscillator such as a klystron or a magnetron and controls its duration by a modulator.
- A waveguide that links the transmitter and the antenna.
- A duplexer that serves as a switch between the antenna and the transmitter or the receiver for the signal when the antenna is used in both situations.
- A receiver. Knowing the shape of the desired received signal (a pulse), an optimal receiver can be designed using a matched filter.
- A display processor to produce signals for human-readable output devices.
- An electronic section that controls all those devices and the antenna to perform the radar scan ordered by software.
- A link to end-user devices and displays.



Info

Radar components

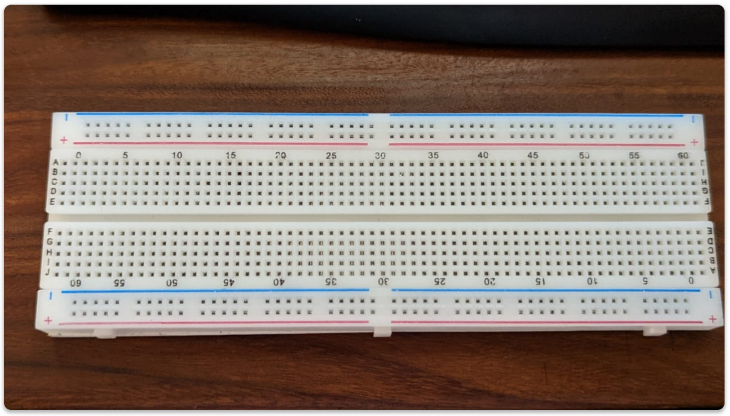
II- Overview of our project: the mobile radar

We're setting out to build a mobile radar system from the components we have in our Arduino kit. We will have to interface the Ultrasonic sensor HC-SR04 and a small Servo Motor for rotating the sensor, with Arduino.

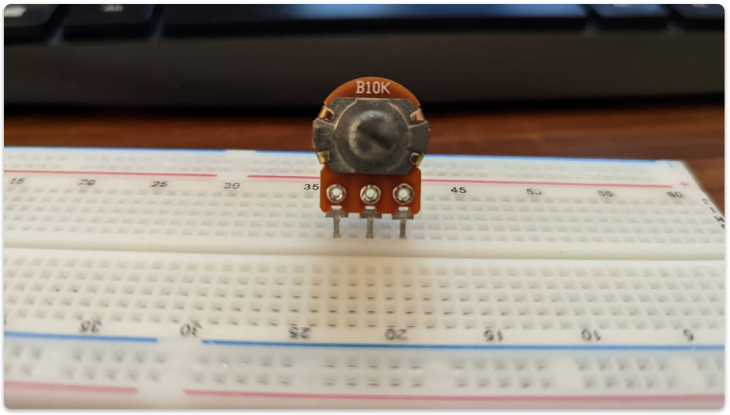
For this project we will use the following hardware:

- Arduino UNO R3 board $\times 1$;

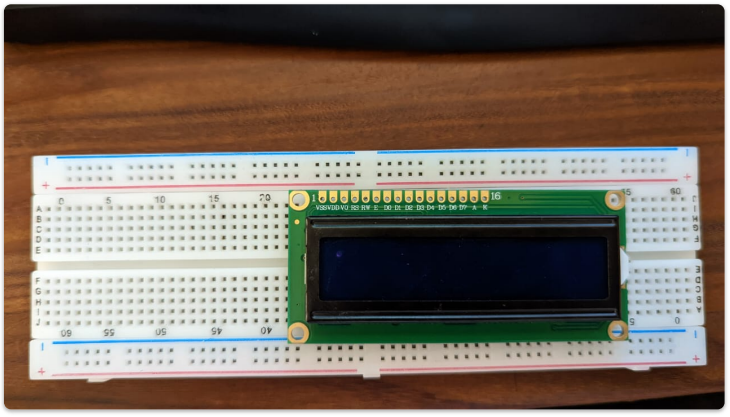
- Breadboard $\times 1$;



- $10k\ \Omega$ rotary potentiometer $\times 1$;

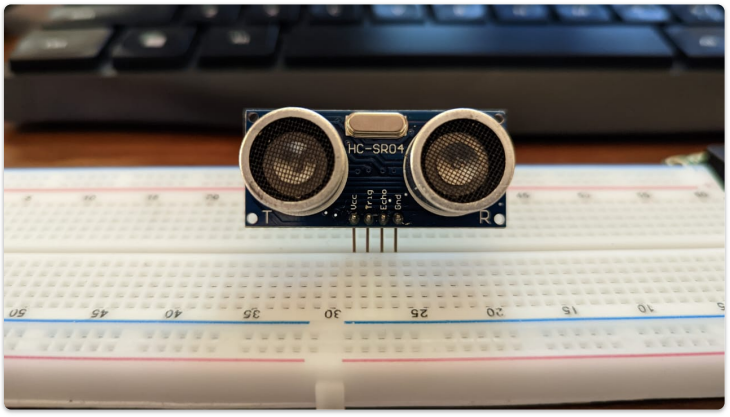


- LCD 16×1;



- 9 g servo motor $\times 1$;

- Ultrasonic sensor HC-SR04 ×1;



III- Overview of our components

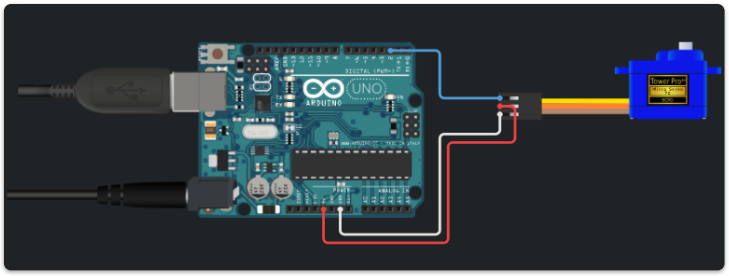
Components-wise, this project is not heavy, though two of them might seem intimidating at first glance. We will go over them, to explain how they work and how we can use them in our circuits.

1. The servo motor:

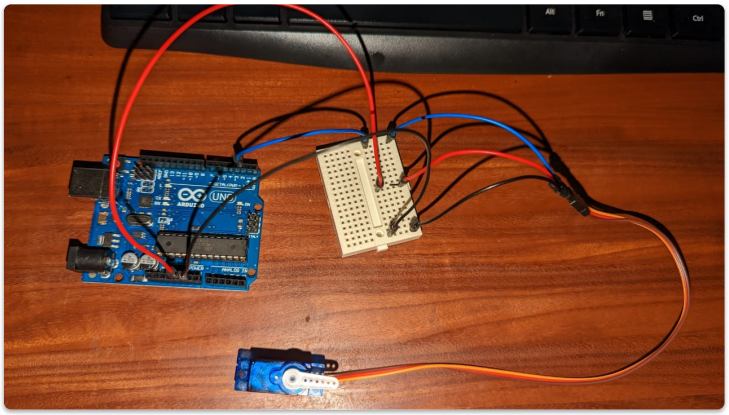
The servo motor is an electro-mechanical component with three pins, PWM/signal (orange), VCC(red), GND(brown); it can rotate approximately 180° (90° in each direction).

a) Wiring to Arduino:

The typical connection between an Arduino R3 board and a servo motor is as follows:



With the VCC connected to 5 V, GND to GND and PWM to a digital pin(D2) in this case; we can build this circuit on a breadboard to have more visibility.



b) Code and manipulation:

This component's role in our project is to rotate the ultrasonic sensor when placed under it. This will allow our radar to cover a larger area. To rotate our sensor, we will make the servo's shaft sweep back and forth. It is achieved with this code:

```ino
#include <Servo.h> //we use the servo library


Servo myServo //create an object called 'myServo'
int i = 0; // setting servo starting point to 0


void setup(){
    myServo.attach(2); //link servo PWM to pin 2
}



void loop(){
    for(int i = 0; i <= 180; i++){ // this will rotate the servo 180 degrees
           myServo.write(i);
        delay(30);
    }
    for(int i = 180; i >= 0; i--){ // this will make the servo go back to its initial position
        myServo.write(i);
        delay(30);
    }
}
```

With a 180° coverage, we have hopefully a sufficient proof of concept while saving some processing power and energy.

2. The ultrasonic sensor:

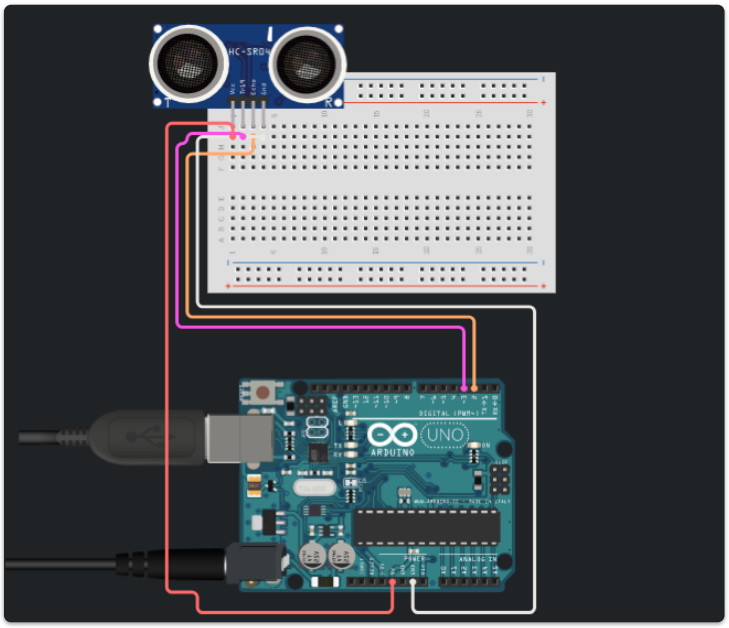
The HC-SR04 ultrasonic distance sensor provides 2 cm to 400 cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3 mm. It includes an ultrasonic transmitter, a receiver, and a control circuit(to prevent inconsistent data). We have 4 pins to manage:

- VCC(power);
- Trig (trigger, with 10 μs TTL(transistor-transistor logic) pulse);
- Echo (receiver, input TTL lever signal and the range in proportion);
- GND (0 V).

You only need to supply a short 10 μs pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo.

a) Wiring to Arduino:

Here we a have a typical connection from the R3 board to the sensor.



with this connection, the only way we get the information taken in by the sensor is through the serial monitor.

b) Code and manipulation:

This is the code we will use to check our sensor:

```ino
int trigPin = 7;
int echoPin = 8;
int travelTime;

void setup (){
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
        Serial.begin(9600); //begin serial communication
}



void loop(){
    //activating the trigger
    digitalWrite(trigPin, LOW);
    delayMicroseconds(10);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    //listening to the echo
    travelTime = pulseIn(echoPin, HIGH);
    delay(25);

    //printing the travel time between the sensor and the object
    Serial.printl(travelTime);
}
```

For an object placed 5 cm away from the sensor, the travel time is around 220 and 230 ms, give or take one to three microseconds. We can check the serial monitor:

From that, and with the principles we went over earlier, we can have the serial monitor display the distance in centimeters with this formula:

$$D = \frac{T_t \times 0.034}{2}$$

where D is the distance to the target, T_t the travel time.

Our code then becomes:

```ino
int trigPin = 7;
int echoPin = 8;
int travelTime;
float distanceToTarget;


void setup(){
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
        Serial.begin(9600);
}



void loop(){
    digitalWrite(trigPin, LOW);
    delayMicroseconds(10);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    travelTime = pulseIn (echoPin, HIGH);
    delay(25);

    //distance calculation
    distanceToTarget(travelTime * 0.034)/2;
    Serial.println(distanceToTarget)
}
```

The serial monitor outputs:

We have approximately 5 *cm* between our target and the sensor, so everything is working as intended.

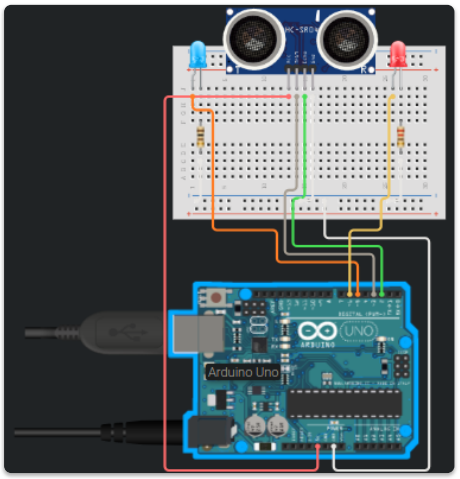
IV- Our prototypes

1. LED activated:

We might want a way to read the signal without using the serial monitor; for that, we devised a rather elementary solution using LEDs. This prototype will work such that if an object is within a certain distance (we called it the **critical distance**) of the sensor, 30 *cm*, a blue LED will light up. Past a certain point, 10 *cm* roughly, a red LED will light up. If the object is not in that 30 *cm* range, there will be no light.

a) Circuit and schematics:

Here is the circuit we will build:



Being that we added two LEDs to the circuit, we will not go over the wiring again.

b) Code and manipulation:

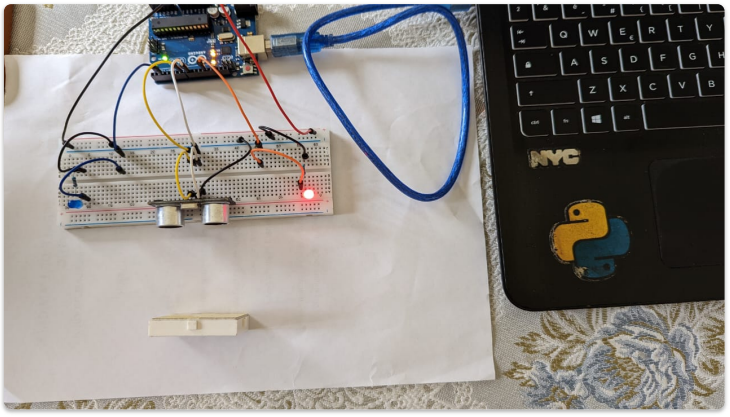
For better code readability, going forward we will use the library HCS04 by gamegine (see links).

```ino
#include <HCSR04.h>


HCSR04 hc (2,12); //trigger first, echo second


int red = 10;
int blue = 6;
float D;

void setup (){
    pinMode (blue, OUTPUT);
    pinMode(red, OUTPUT);
        Serial.begin(9600); //so we can check the exact distance on the serial monitor
}



void loop(){
    D = hc.dist();
    Serial.println (D);

    for (D > 3.00; D ≤ 30.00; D++){ //objects detected within this range will activate the
red LED
        digitalWrite(red, HIGH);
```

```
        digitalWrite(blue, LOW);
    }

    for (D > 30.00; D = 100.00; D++){ //objects detected within this range will activate the
blue LED
        digitalWrite(red, LOW);
        digitalWrite(blue, HIGH);
    }

    if (D > 100.00){ // beyond 100 meters, the object is out of our "critical range"
        digitalWrite (red, LOW);
        digitalWrite (blue, LOW);
        }
}
```

c) Results and limits:

This prototype works exactly as intended. Here, we've set an obstacle 7 *cm* away:



Then 34 *cm* away:

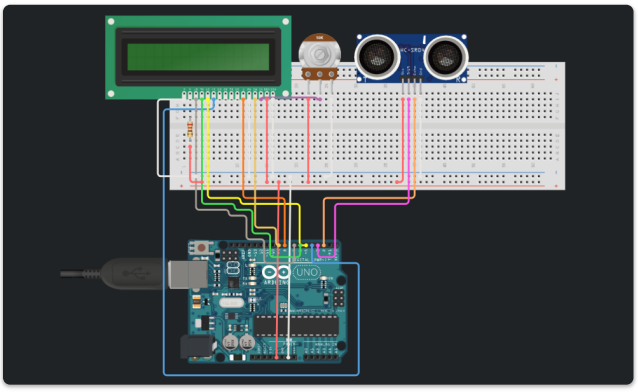
If the obstacle is 1 *m* away, there will be no light. Though it works it does not allow us to know exactly how far the object is. We need some form of display, to have exact distances.

2. Distances on an LCD:

With this prototype, we will have the distance displayed on an LCD. No light is needed for this since we will have the exact distance in centimeters on the display.

a) Circuit and schematics:

We will have the following circuit:



The wiring for the LCD is the following:

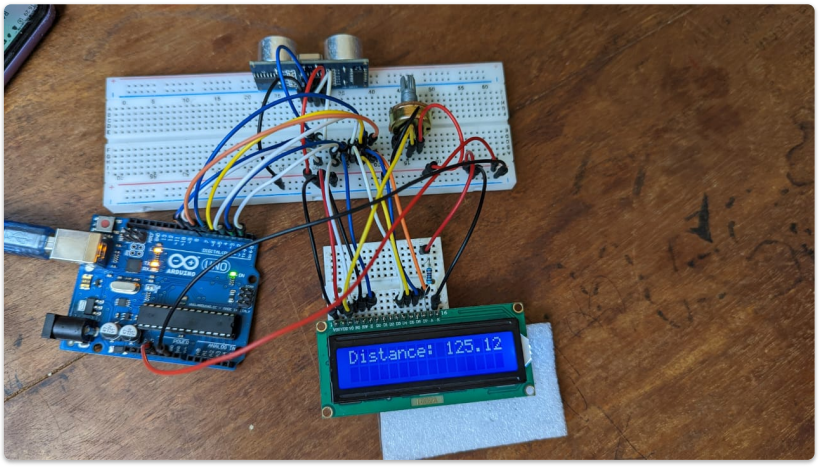
- VSS to 5 V;
- VDD to GND;
- VO to the potentiometer's wiper;
- RS to a digital pin;
- R_W (read-write) to GND;
- E to a digital pin;
- D4 through D7 to digital pins;
- A (LED +) to 5 V;
- K (LED -) to GND.

b) Code and manipulation:

```ino
#include <LiquidCrystal.h>
#include <HCSR04.h>


LiquidCrystal lcd (1,2,4,5,6,7); // creation of the LCD object
HCSR04 (9,10);


void setup(){
    lcd.begin(16,2);
}


void loop (){
    lcd.setCursor (0, 0);
    lcd.print ("Distance: ");
    lcd.print (hc.dist());
    lcd.print (" cm");
    delay (10);
}
```

c) Results and limits:

With this, we can see the distance to the nearest object clearly, without the need to know how to use the serial interface.



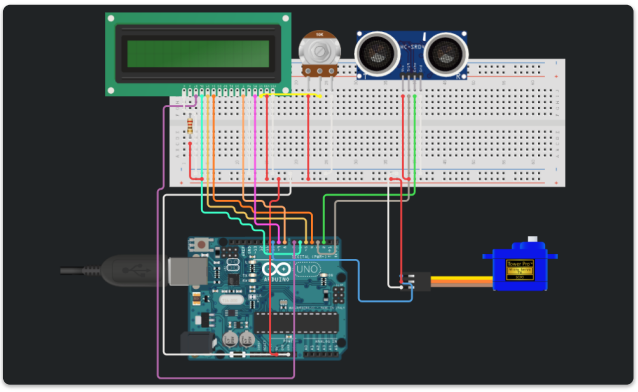
What limits this prototype, is that it can only ”see” what is in front of it. For the device to be efficient, it needs to move and cover more ground when ”watching”. We can use a motor to rotate the sensor.

3. Rotating sensor:

For this prototype, we will have the sensor mounted to a 9 g servo motor, to help us get more information.

a) Circuit and schematics:

We want a circuit such as this one:



b) Code and manipulation:

```ino
#include <HCSR04.h>
#include <Servo.h>
#include <LiquidCrystal.h>


Servo myServo;
HCSR04 hc (10, 11);
LiquidCrystal lcd (1, 2, 4, 5, 6, 7);


int i = 0;


void setup (){
```

```
    lcd.begin(16, 2);
    myServo.attach(8);
}
void loop (){
    lcd.setCursor (0, 0);

    for (i = 0; i <=180; i++){
        myServo.write(i);
        delay (30);
        lcd.print("Distance: ");
        lcd.print(D);
        lcd.print(" cm");
        delay (10);
    }

    for (i = 180; i >= 0; i--){
        myServo.write (i);
        lcd.print ("Distance: ");
        lcd.print (D);
        lcd.print (" cm");
        delay (10);
    }
}
```

c) Results and limits:

”prototype3” is not created yet. Click to create.

The limiting factor for this device is that when only referring to the screen, we are unable to determine where each object is coming from.

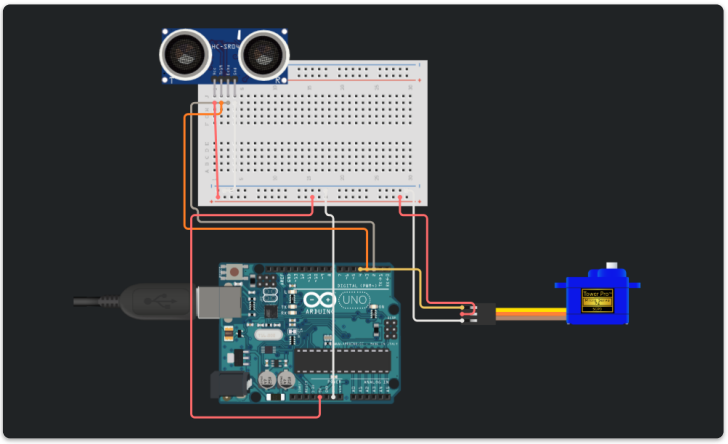
V- Our mobile radar

1. Presentation:

This system comes from taking into account the limits of each prototype.

2. Circuit and schematics:

The circuit we want to build is similar to the ones we’ve already encountered during our research.



3. Code:

For precision's sake, we will have to stop using the sensor's library, as it proved fairly imprecise and inconsistent.

```ino
// Includes the Servo library
#include <Servo.h>


// Defines Trig and Echo pins of the Ultrasonic Sensor
const int trigPin = 7;
const int echoPin = 8;


// Variables for the duration and the distance
long duration;
int distance;


Servo myServo; // Creates a servo object for controlling the servo motor


void setup() {
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
    Serial.begin(9600);
    myServo.attach(5); // Defines on which pin is the servo motor attached
}


void loop() { // rotates the servo motor from 5 to 175 degrees
    for(int i = 5; i <= 175; i++){
        myServo.write(i);
        delay(30);
        distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree
        Serial.print(i); // Sends the current degree into the Serial Port
        Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
        Serial.print(distance); // Sends the distance value into the Serial Port
        Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
    }

    // Repeats the previous lines from 165 to 15 degrees
    for(int i=175; i>5; i--){
        myServo.write(i);
        delay(30);
        distance = calculateDistance();
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
    }
}


// Function for calculating the distance measured by the Ultrasonic sensor


int calculateDistance(){
```

```
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel
time in microseconds
    distance= duration*0.034/2;
    return distance;
}
```

4. The radar interface:

We will receive the values for the angle and distance measured by the sensor from the Arduino board into the processing IDE using the SerialEvent() function which reads data from the serial port.

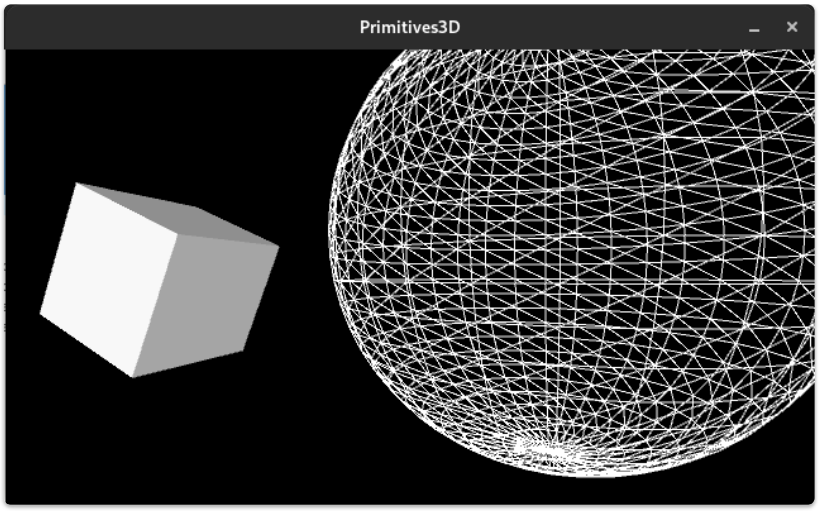
a) The processing IDE:

Processing is a free, java-based, object-oriented language and IDE built for the electronic arts and visual design communities.

With this tool we can draw figures able to interact with our Arduino code. Here is an example:

```language-processing
size (640, 360, P3D);
background(0);
lights();


noStroke();
pushMatrix();
translate(130, height/2, 0);
rotateY(1.25);
rotateX(-0.4);
box(100);
popMatrix();


noFill();
Stroke(255);
pushMatrix();
translate(500, height*0.35, -200);
sphere(280);
popMatrix();
```

This yields:



We will attempt to reproduce the well-known radar interface.

b) Code:

Since the Java language might not be everyone's strong suit, we tried commenting on our code as much as possible to ensure the reader's comprehension.

```
import processing.serial.*; // imports library for serial communication language-processing
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;


Serial myPort; // defines Object Serial


// defines variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;


void setup() {
    size (1366, 768); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
    smooth();
    myPort = new Serial(this,"/dev/ttyACMX", 9600); // starts the serial communication
    myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'.
So actually it reads this: angle,distance.
}


void draw() {
    fill(0, 255, 0);

    // simulating motion blur and slow fade of the moving line
    noStroke();
    fill(0,4);
    rect(0, 0, width, height-height*0.065);
```

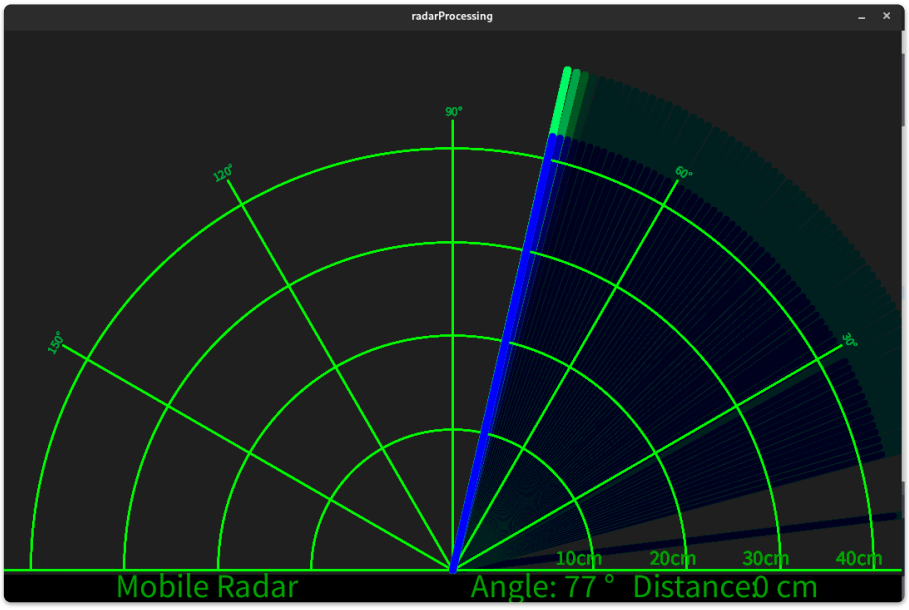
```
    fill(0, 255, 0);

        // calls the functions for drawing the radar
    drawRadar();
    drawLine();
    drawObject();
    drawText();
}

void serialEvent (Serial myPort) { // starts reading data from the Serial Port
    data = myPort.readStringUntil('.'); // reads the data from the Serial Port up to the
character '.' and puts it into the String variable "data".
    data = data.substring(0,data.length()-1);

        index1 = data.indexOf(","); // find the character ',' and puts it into the variable
"index1"
    angle= data.substring(0, index1); // read the data from position "0" to position of the
variable index1 or thats the value of the angle the Arduino Board sent into the Serial Port
    distance= data.substring(index1+1, data.length()); // read the data from position
"index1" to the end of the data pr thats the value of the distance

    // converts the String variables into Integer
    iAngle = int(angle);
    iDistance = int(distance);
}



void drawRadar() {
    pushMatrix();
    translate(width/2, height-height*0.074); // moves the starting coordinates to new
location
    noFill();
    strokeWeight(2);
    stroke(0, 255, 0);

    // draws the arc lines
    arc(0, 0, (width-width*0.0625), (width-width*0.0625), PI, TWO_PI);
    arc(0, 0, (width-width*0.27), (width-width*0.27), PI, TWO_PI);
    arc(0, 0, (width-width*0.479), (width-width*0.479), PI, TWO_PI);
    arc(0, 0, (width-width*0.687), (width-width*0.687), PI, TWO_PI);

    // draws the angle lines
    line(-width/2,0,width/2,0);
    line(0, 0, (-width/2)*cos(radians(30)), (-width/2)*sin(radians(30)));
    line(0, 0, (-width/2)*cos(radians(60)), (-width/2)*sin(radians(60)));
    line(0, 0, (-width/2)*cos(radians(90)), (-width/2)*sin(radians(90)));
    line(0, 0, (-width/2)*cos(radians(120)), (-width/2)*sin(radians(120)));
    line(0, 0, (-width/2)*cos(radians(150)), (-width/2)*sin(radians(150)));
    line((-width/2)*cos(radians(30)), 0, width/2, 0);
    popMatrix();
}



void drawObject() {
    pushMatrix();
    translate(width/2, height-height*0.074); // moves the starting coordinates to new
location
    strokeWeight(9);
    stroke(0, 0, 255);
```

```
    pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the
sensor from cm to pixels
    // limiting the range to 40 cm
       if(iDistance<40){// draws the object according to the angle and the distance
       line(pixsDistance*cos(radians(iAngle)), -pixsDistance*sin(radians(iAngle)), (width-
width*0.505)*cos(radians(iAngle)), -(width-width*0.505)*sin(radians(iAngle)));
    }
    popMatrix();
}

void drawLine() {
    pushMatrix();
    strokeWeight(9);
    stroke(0, 250, 100);
    translate(width/2, height-height*0.074); // moves the starting coordinates to new
location
    line(0, 0, (height-height*0.12)*cos(radians(iAngle)), -(height-
height*0.12)*sin(radians(iAngle))); // draws the line according to the angle
    popMatrix();
}



void drawText() { // draws the texts on the screen
    pushMatrix();
    if(iDistance>40) {
          noObject = "Out of Range";
    } else {
       noObject = "In Range";
    }

    fill(0,0,0);
    noStroke();
    rect(0, height-height*0.0648, width, height);
    fill(98,245,31);
    textSize(25);
    text("10cm", width-width*0.3854, height-height*0.0833);
    text("20cm", width-width*0.281, height-height*0.0833);
    text("30cm", width-width*0.177, height-height*0.0833);
    text("40cm", width-width*0.0729, height-height*0.0833);

    textSize(40);
    text("Mobile Radar", width-width*0.875, height-height*0.0277);
    text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
    text("Distance: ", width-width*0.26, height-height*0.0277);
    if(iDistance<40) {
          text("       " + iDistance +" cm", width-width*0.225, height-height*0.0277);
    }

    textSize(25);
    fill(98,245,60);
    translate((width-width*0.4994)+width/2*cos(radians(30)), (height-height*0.0907)-
width/2*sin(radians(30)));
    rotate(-radians(-60));
    text("30°", 0, 0);
    resetMatrix();

    translate((width-width*0.503)+width/2*cos(radians(60)), (height-height*0.0888)-
width/2*sin(radians(60)));
    rotate(-radians(-30));
```

```
    text("60°", 0, 0);
    resetMatrix();
    translate((width-width*0.507)+width/2*cos(radians(90)), (height-height*0.0833)-
width/2*sin(radians(90)));
    rotate(radians(0));
    text("90°", 0, 0);
    resetMatrix();
    translate(width-width*0.513+width/2*cos(radians(120)), (height-height*0.07129)-
width/2*sin(radians(120)));
    rotate(radians(-30));
    text("120°", 0, 0);
    resetMatrix();
    translate((width-width*0.5104)+width/2*cos(radians(150)), (height-height*0.0574)-
width/2*sin(radians(150)));
    rotate(radians(-60));
    text("150°", 0, 0);
    popMatrix();
}
```

c) Interface results:

This is the interface we get:



5. Demonstration:

See PowerPoint presentation.

Ω

#embedded_systems , #ES_project

links and sources:

- WIKIPEDIA: RADAR (https://en.wikipedia.org/wiki/Radar)
- How does RADAR work? | James May Q&A | Head Squeeze (https://www.youtube.com/watch?v=ywDE57CtaTM)
- History of radar (https://www.britannica.com/technology/radar/History-of-radar)
- Radio wave (https://en.wikipedia.org/wiki/Radio_propagation)
- FMCW Radar for Autonomous Vehicles | Understanding Radar Principles (https://www.youtube.com/watch?v=-N7A5CIi0sg)
- Radar engineering details (https://en.wikipedia.org/wiki/Radar_engineering_details)
- Arduino Radar Project (https://howtomechatronics.com/projects/arduino-radar-project/)
- Ultrasonic Sensor HC-SR04 with Arduino Tutorial (https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6)
- Servo motor SG90 data sheet (http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)
- Ultrasonic Ranging Module HC - SR04 (https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf)
- Simple Arduino and HC-SR04 Example (https://www.instructables.com/Simple-Arduino-and-HC-SR04-Example/)
- Arduino Tutorial 53: Understanding and Connecting the HC-SR04 Sensor(https://toptechboy.com/arduino-tutorial-53-understanding-and-connecting-the-hc-sr04-sensor/)
- Arduino lib for HCSR04 ultrasonic sensor (https://github.com/gamegine/HCSR04-ultrasonic-sensor-lib)
- Arduino Radar Project (https://create.arduino.cc/projecthub/electronicslife/arduino-radar-project-bb7d69)