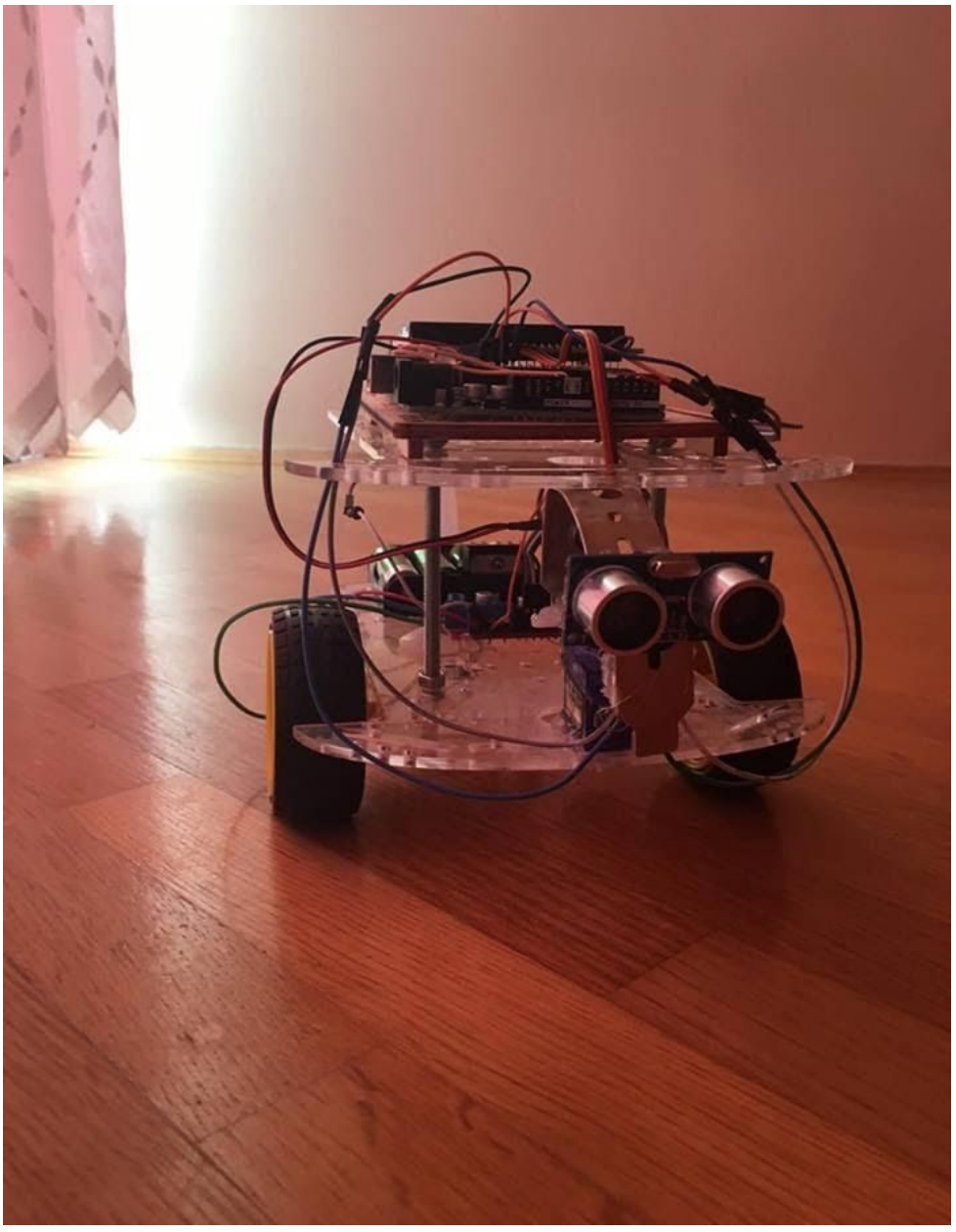
# HARDWARE CONTROL (with Arduino)

## **The measuring device: TIPO**

Blend Hamiti & Premt Cara - 25th of May, 2016



# 1.Project abstract

The distance measuring robot - TIPO - is an Arduino based project with the goal of helping architects measure areas of weirdly shaped rooms quickly and easily.

# 2. Project Objectives

**What's an architect?**
An architect is a person who is between an artist and an engineer, he is the one who draws and designs buildings and also thinks about the technical part.

**What does an architect do?**
Before constructing a building an architect needs to draw a plan of the building, then the building is build by a construction company which follows the direction of the plan, the architect will closely supervise the construction and make sure that the building is build according to the plan. But before drawing the plan he has to measure the dimensions of the place where he is going to build the building and so It goes on. Being an architect it's not just about designing building, it's also about making our houses and our workplaces comfortable, functional and beautiful and these are included in interior design. And that's where our idea kicks in, it suggests a more practical way to deal with the initial problem of interior design- MEASURING THE AREA OF THE ROOM.
And what we are about to present you is a programmed hardware that is related to these things. In fact it's a measuring vehicle. It measures the surfaces area of a room that's in a square or rectangular shape. It helps architects measure the rooms bottom surface faster, within a minute.
We've been working with this with an arduino board, we've taken some sensors, we've mixed some projects, we've programmed in a different way, a more complex one and hopefully it'll do its job.
This projects is just about measuring the bottom surface, but later if it would be possible we will try to make it for measuring any room and for volume measurement, but for now unfortunately the project is only about measuring the bottoms surface.

## THE PROJECT CONSISTS OF :

- Arduino Uno board
- H-bridge motorboard
- Breadboard
- Usb cable
- Jumper Wires (male-male, male-female, female-female)
- External Battery
- HCR-04 Distance sensor
- LCD Display
- Vehicle Kit (3xWheels, 2xPlatforms, 2xElectromotors, 4x1.5V-Batteries)

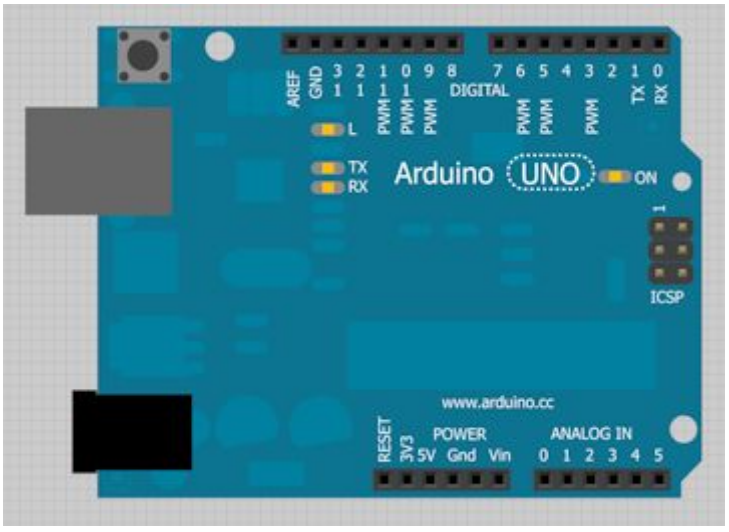
## Programming The Arduino

This is an introduction of the progamming side of this project. It has enough in information that can be applied to most Arduino projects. By the end of it you will know the basics about how these sensors, motors and other devices connected to the Arduino.

## The Arduino Pins

The Arduino board communicates with connected devices via its input and output pins, so I'll quickly go over what these pins are.

Here is a diagram of a typical Arduino board.




On the left side you have the USB port (grey box) and the power input jack (black box). During testing I will be powering my Arduino with a USB cable, but the final robot will receive power from a battery box connected to the power input jack.

On the top, from right to left there are 14 pins labeled **0** to **13**. These are the **digital pins**. These pins can be individually configured as inputs or outputs, meaning that digital data can be read or written from connected devices on these pins. Since these pins are digital, they have only two possible states, **HIGH** and **LOW**.

Some of the digital pins have preassigned functions. Pins 0 and 1 are also labeld RX and **TX**respectively. These are used by the serial communication hardware to send and receive data. Pin 13 has a **LED** attached to it on most Arduino boards, so it is a convenient pin to send simple visual information out to the real world. The pin 13 LED is located below the pin 13 itself, labeled with the letter **L** in the diagram above. Pins 3, 5, 6, 9, 10 and 11 are marked with a ~ or a **PWM**label, short for **pulse widht modulation**. These pins are capable of producing simulated analog output over a digital line. These can be used, for example, to light a LED at different levels of intensity.

The next pin on the top from right to left is **GND**, short for ground. The ground is what closes a circuit and allows the electric current to flow uninterrupted. Connected devices can connect their own GND pin here.

The next pin on the top left is labeled **AREF**, short for analog reference. This pin is rarely used, it tells the Arduino how to configure the voltage range of the analog pins.

Some Arduino boards have more pins to the left of **AREF**. Since these are not in all boards I will not discuss them here.

On the bottom right we have six pins labeled 0 to 5. These are the **analog input pins**. Unlike the digital pins which have only two possible states, an analog pin can have 1024 possible states, according to the voltage applied to it. Typically the voltage range goes up to 5V, but the range can be changed by applying the desired maximum voltage to the **AREF** pin. Also unlike the digital pins which can be configured as inputs or outputs, the analog pins can only be inputs.

An interesting property of the analog pins is that they can be used as digital pins as well, with pin numbers 14 to 19. You will see later that an important part of any project is allocating pins to components. If you are running short of digital pins keep in mind that you have six more available to you.

Continuing on the bottom from right to left we find a pin labeled **VIN**. This pin provides direct access to the voltage provided by the power supply. So for example, if you power your Arduino with a 9V power supply through the power jack, then this pin gives you 9V.

The next two pins are labeled **GND** and are two more ground pins, exactly like the one in the top row. They are here just for convenience.

The next two pins are labeled **5V** and **3.3V** and just return those voltages, regardless of what the voltage of the power supply is.

Next we have the **RESET** pin. When this pin is connected to **GND** the board will reset. So this is a handy pin to build an external reset button.

As with the top row of pins, some boards have a more pins on the bottom left that I will not cover here. If you are interested in finding out what extra pins in your board do you should consult the documentation for your board.

**Our projects has two functions:**

- Measuring surfaces area
- Obstacle Avoidance

## Firstly we need to include the libraries we will be working with.

```
#include <NewPing.h>
#include <Servo.h> // Load servo library
#include <LiquidCrystal.h> //Load Liquid Crystal Library
```

## Then we need to define display pins.

```
//Define display pins
#define pinDistanceOut 11 //Sensor Trip pin connected to Arduino pin 12
#define pinDistanceIn 10 //Sensor Echo pin connected to Arduino pin 13


//Define engine pins
#define pinEngineLeftRun A4
#define pinEngineLeftBrake A3
#define pinEngineRightRun A2
#define pinEngineRightBrake A1

const int myServoPin = 12;

//Define global variables
int robotLength=20;
int defaultSpeed = 50;
int defaultDelay = 100; //default loop delay in ms
int minDelay = 20; // minimal delay during operations
long wallDistance = 10; // minimal distance to keep from the wall in cm
bool awayFromWall = false;

int activeWall = 0;
long minDistance = 10000;
long maxDistance = 0;

long activeDistance = 0;
long lastDistance = 0;
bool increasingDistance = false;
```

```
bool firstRun = true;

//wall distances
long wall1Max = 0;
long wall1Min = 0;
long wall2Max = 0;
long wall2Min = 0;
long wall3Max = 0;
long wall3Min = 0;
long wall4Max = 0;
long wall4Min = 0;

int numberOfPoints = 35;
int delayPerRotation = 100;
//int points[] = {5,4,5,6,8,7,6,7,4,3,2,3,4,7,6,5,6,7,8,7,5};
int points[100];
int A = 0;
int B = 0;
```

## We need to setup the functions:

```
void setupDisplay()
{
  LCD.begin(16,2); //Tell Arduino to start your 16 column 2 row LCD
  LCD.setCursor(0,0);  //Set LCD cursor to upper left corner, column 0, row 0
  LCD.print("Target Distance:");  //Print Message on First Row
}

void updateDistanceInDisplay(long distance)
{
  LCD.setCursor(0,1);  //Set cursor to first column of second row
  LCD.print("                "); //Print blanks to clear the row
  LCD.setCursor(0,1);   //Set Cursor again to first column of second row
  LCD.print(distance); //Print measured distance
  LCD.print(" cm");  //Print your units.
  Serial.println(distance);

}
```

## Then we need to setup up the functions for the animation/movement.

```
void moveAwayFromTheWall(long dist)
{
  if(awayFromWall==true && dist>=wallDistance){
    stopEngines();
    return;
  }
  if(dist<wallDistance && awayFromWall==false){
    moveBackward();// start moving back
    awayFromWall=true;
  }
}
```

## Moving forward and backward

```
void moveBackward(){
  leftEngine(false);
  rightEngine(false);
  //delay(20);
}

void moveForward(){
  leftEngine(true);
  rightEngine(true);
  //delay(20);
}
```

## Engine Breaks

```
void stopEngines(){
  digitalWrite(pinEngineLeftRun, LOW);
  digitalWrite(pinEngineRightRun, LOW);
  digitalWrite(pinEngineLeftBrake, LOW);
  digitalWrite(pinEngineRightBrake, LOW);
}

void leftEngine(bool forward){
  if(forward){
    digitalWrite(pinEngineLeftRun, HIGH);
    digitalWrite(pinEngineLeftBrake, LOW);
  }else{
    digitalWrite(pinEngineLeftRun, LOW);
    digitalWrite(pinEngineLeftBrake, HIGH);
  }

}
```

```
void stopEngines(){
  digitalWrite(pinEngineLeftRun, LOW);
  digitalWrite(pinEngineRightRun, LOW);
  digitalWrite(pinEngineLeftBrake, LOW);
  digitalWrite(pinEngineRightBrake, LOW);
}
```

**Moving the robot.**

```
void leftEngine(bool forward){
  if(forward){
    digitalWrite(pinEngineLeftRun, HIGH);
    digitalWrite(pinEngineLeftBrake, LOW);
  }else{
    digitalWrite(pinEngineLeftRun, LOW);
    digitalWrite(pinEngineLeftBrake, HIGH);
  }

void rightEngine(bool forward){
  if(forward){
    //analogWrite(rightEnablePin, defaultSpeed);
    digitalWrite(pinEngineRightRun, HIGH);
    digitalWrite(pinEngineRightBrake, LOW);
  }else{
    digitalWrite(pinEngineRightRun, LOW);
    digitalWrite(pinEngineRightBrake, HIGH);
  }
}
```

## Rotation left/right.

```
void rotateLeft(){
  rightEngine(true);
  leftEngine(false);
  //delay(minDelay);
}

void rotateRight(){
  leftEngine(true);
  rightEngine(false);
  //delay(minDelay);
}
```

## Starting the function

```
void setup() {
  //initialize distance measuring pins
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  Serial.println("Starting config");
  pinMode(pinDistanceIn, INPUT);
  pinMode(pinDistanceOut, OUTPUT);
  setupDisplay();

  //step away from the wall to min distance
  //moveAwayFromTheWall(10);
}
```

## Calculation of distance...

```
//Calculate distance
long calculatedDistance(){
  long duration;
  digitalWrite(pinDistanceIn, LOW);
  delayMicroseconds(2);
  digitalWrite(pinDistanceIn, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinDistanceIn, LOW);
  duration = pulseIn(pinDistanceOut, HIGH);
  Serial.println("Distance: " + duration);
  return (duration/2) / 29.1;
}

void getPoints() {
  int distance;
  while(elements < numberOfPoints)
  {
    rotateLeft();
    delay(delayPerRotation);
    stopEngines();
    //int distance = calculatedDistance();
    distance = myPing.ping_cm();
    if (distance > 0){
      updateDistanceInDisplay(distance);
      Serial.println(distance);
      points[elements] = distance;
      elements++;
    }
    delay(2000);
  }
}
```

## Calculation loop

```
void loop() {
  if (calculated == false){
    getPoints();
    stopEngines();

    calculateSurface();
    calculated = true;
  }
```

**And the main loop/code goes here**

```
void calculateSurface(){
  int mins[4] = {0, 0, 0, 0};
  int j = 0;
  int prev = 0;
  for(int i=1; i<elements; i++)
  {
    if ((points[i] < points[prev]) && (points[i] < points[i+1]))
    {
      mins[j] = points[i];
      j++;
      prev = i;
    }
    if (points[i] != points[i+1]) {
      prev = i;
    }
    //updateDistanceInDisplay(points[i]);
    delay(200);
  }
  //int S = 0;
  A = mins[0] + mins[2] + robotLength;
  B = mins[1] + mins[3] + robotLength;
  S = A * B;
  updateDistanceInDisplay(S);
}
```

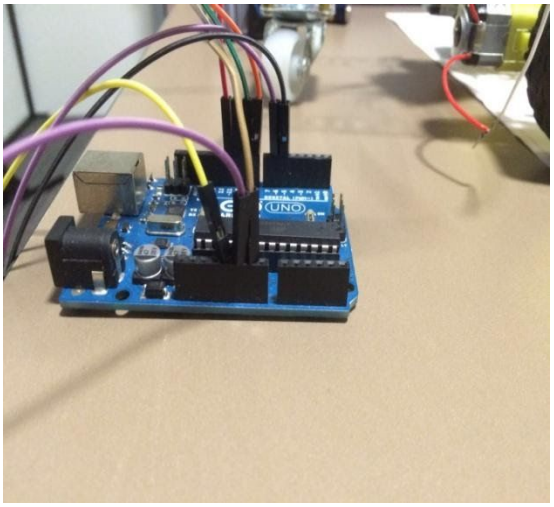
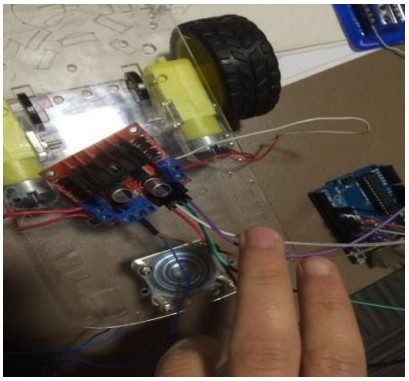
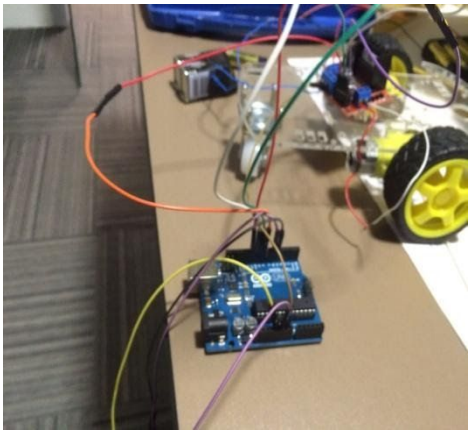
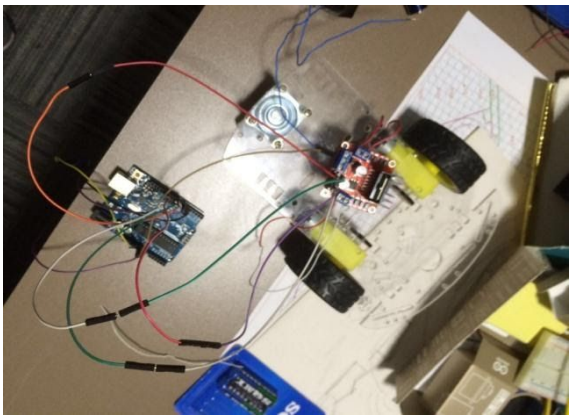
## Obstacle Avoidance

We have added this extra part just because it is fun, it has a great way of moving and its fascinating.
And pretty much this code goes after the robot has finished calculating the area, it is in the **else** statement

```
void loop() {
  if (calculated == false){
    digitalWrite(ledPin, LOW);
    getPoints();
    stopEngines();

    calculateSurface();
    calculated = true;
    digitalWrite(ledPin, HIGH);
  }

  else {
    //Obsatcle Avoidance Code goes here!!!
    obstacleAvoidance();

  }
}
```

And after this is the rest of the code which makes the robot move randomly, going places and avoiding obstacles.

# Assembling the robot

## 3.Project challenges

Since architects need to measure rooms surface they first need to measure one side then go to the other one and in the end they should calculate, thats not hard though but what about a robot doing that job ?!
Thats our idea to solve that problem, to be more exact, to be more reliable.
Not only robot can measure areas but if in any room in any place it can avoid obstacles, it doesn't crash, it doesn't touch anything, for that to be sure, robot moves freely in every direction without having to touch or crash into something.

# 4.Conclusion

It'll come a time when robots will do all the work in order to clarify and simplify humans activity. People will get more lazy, people won't do anything aside controlling the robots. Robots are a great way to finish our jobs, our work but  it also has a bad impact on human physical activity.
Anyway we have made this robot for a particular reason so it can measure distances, areas and not crash. But what we are not able to do is measure the volume of some place, and since we've been working with an arduino board we've had many limitations such as distance, energy and other sensors. The thing is,this robot is not that sophisticated, this robot is just for showing people like a demo that devices like this can be made in order to easement an architect's work. With this robot we want to show that you don't need to measure each side, you don't need to bother doing that, you just order your robot to do it. That's it!

# 5.References

http://blog.miguelgrinberg.com/post/building-an-arduino-robot-part-i-hardware-components
http://www.dummies.com/how-to/content/how-to-measure-distance-with-the-arduino.html