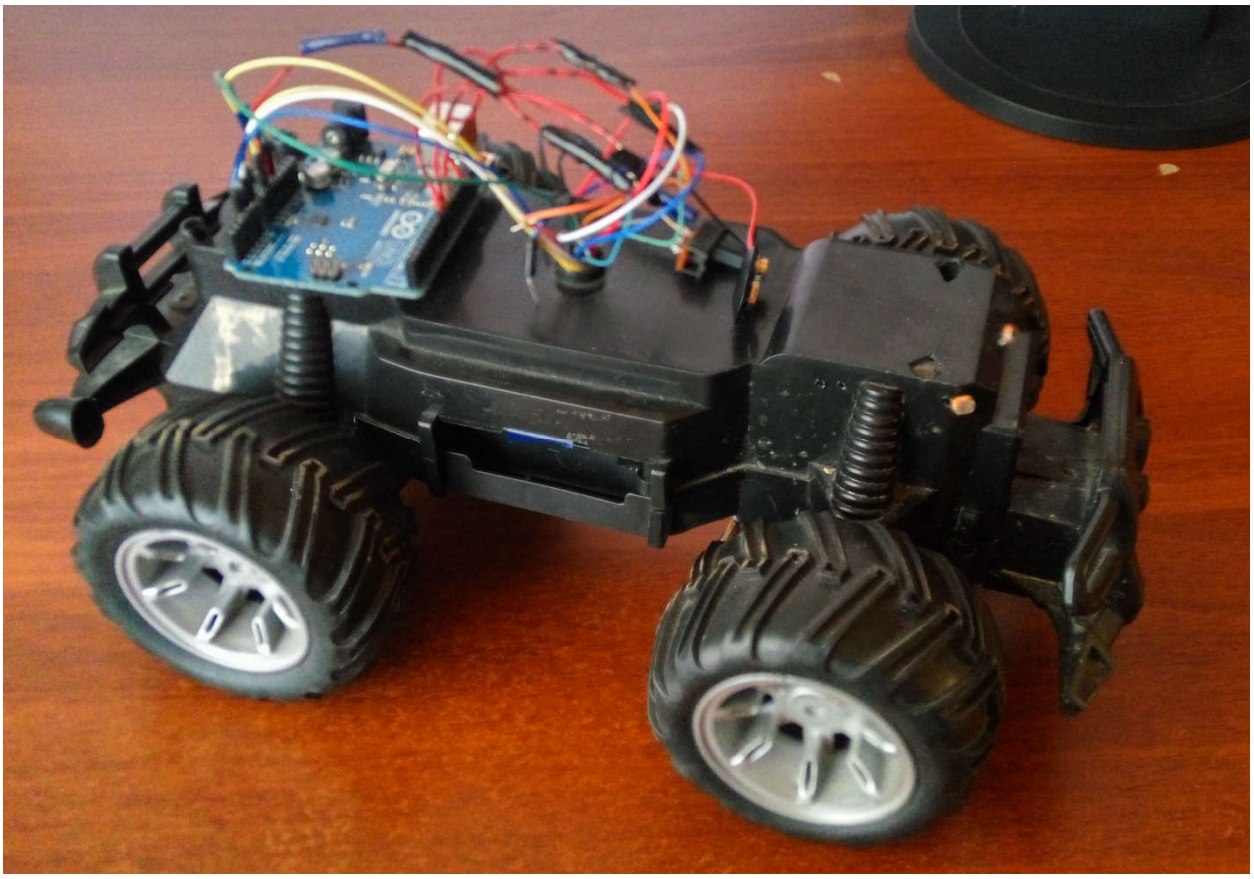


Arduino project description.

The light stalker (2016)

Shakirov Sukhrob Zairovich

Contents

Introduction

The Arduino is a very flexible and convenient micro-controller that can be applied to invent many and many devices. What's more, this device is extremely useful for people with an exceptional creativity.

At the beginning, my idea was to create a device that follows a host and helps to carry a weight. However, due to the lack of device parts the project was changed into more primitive form. Now, the device/car was intended into to chase a source of light directed by a host.

Issues overview

I faced many problems during the construction of the device since parts, that was gotten, were not designed to Arduino. The parts for the device I got from a simple toy car. Hence, I did very painstaking job by solving these issues using a software (Arduino IDE), programming, rather than buying new expensive parts because, in that case, it would be very easy to realize the project. Moreover, it would not be so challenging if I would have bought special Arduino designed machine parts, and a positive effect from the project would not be so sensible.

The idea of the project itself indeed was not that hard in complexity and my intention was to get a valuable experience of working with electrical circuits and Arduino.

During the construction I have encountered several issues concerned mostly with parts that I was using.

1. Environment illumination (incessantly was confusing my photo-resistors, which I have been using as "eyes" for the device. Hence, the car was constantly turning left and right)
2. Driving motor inertia, as well as speed/acceleration.

Components and the Code.

Components:

1. Toy car (Driving motor and ruling mechanism)
2. 3 pieces of Photo-resistor and 3 pieces of resistors (no matter what ohm, in my case they were 220)

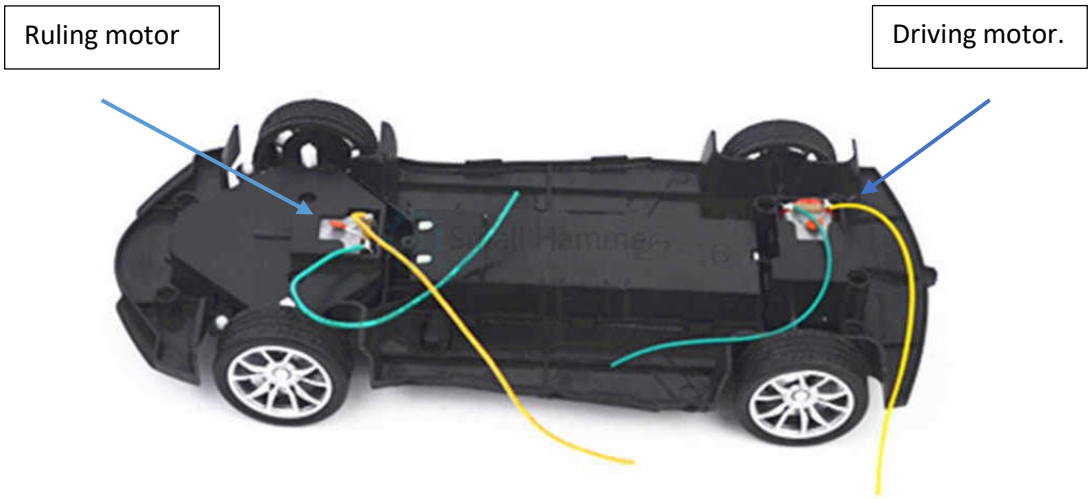
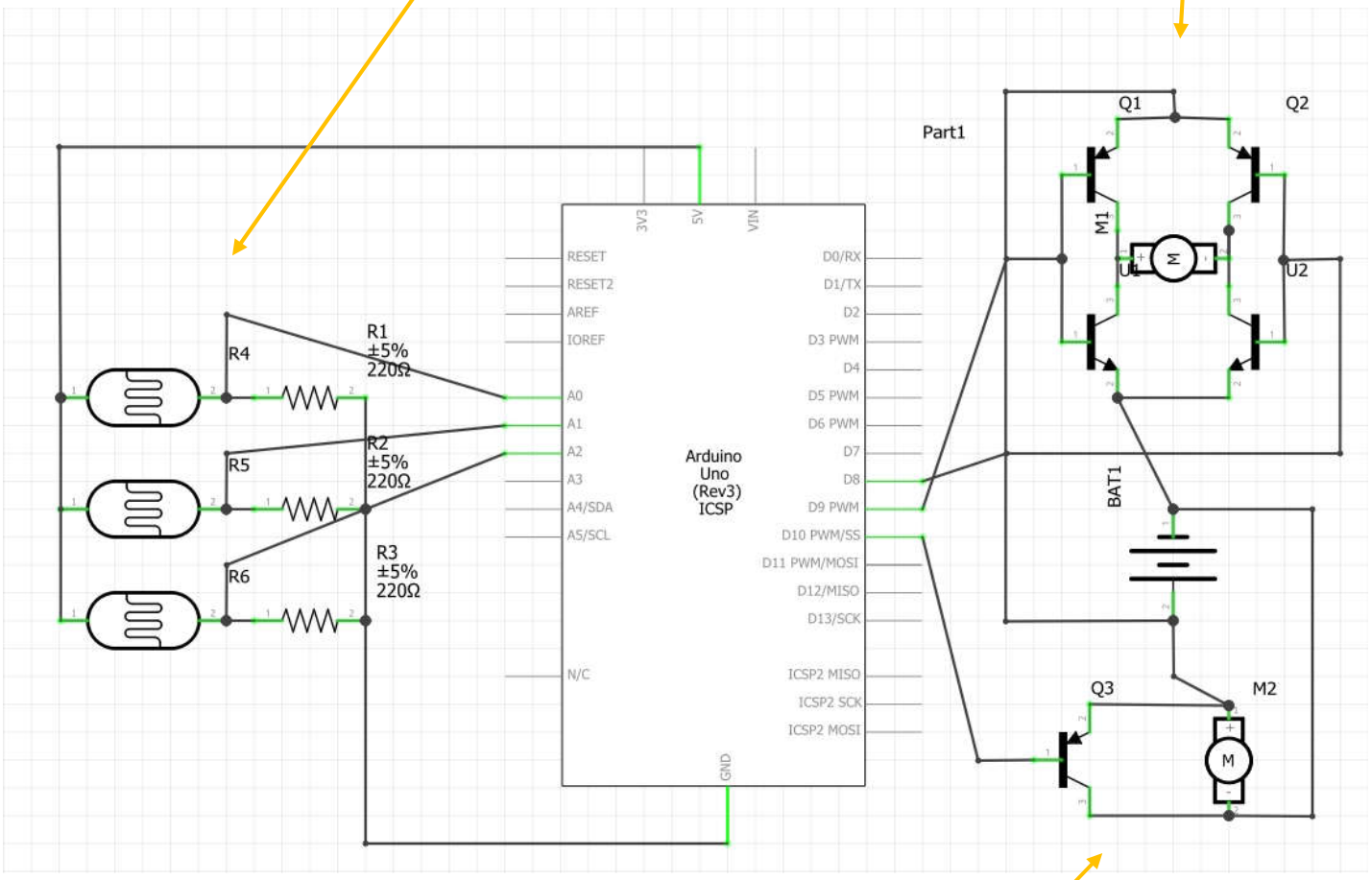


Fig.1. Toy car internals

Fig.2. Photo-resister

Ruling System

Ruling DC motor



Driving DC motor

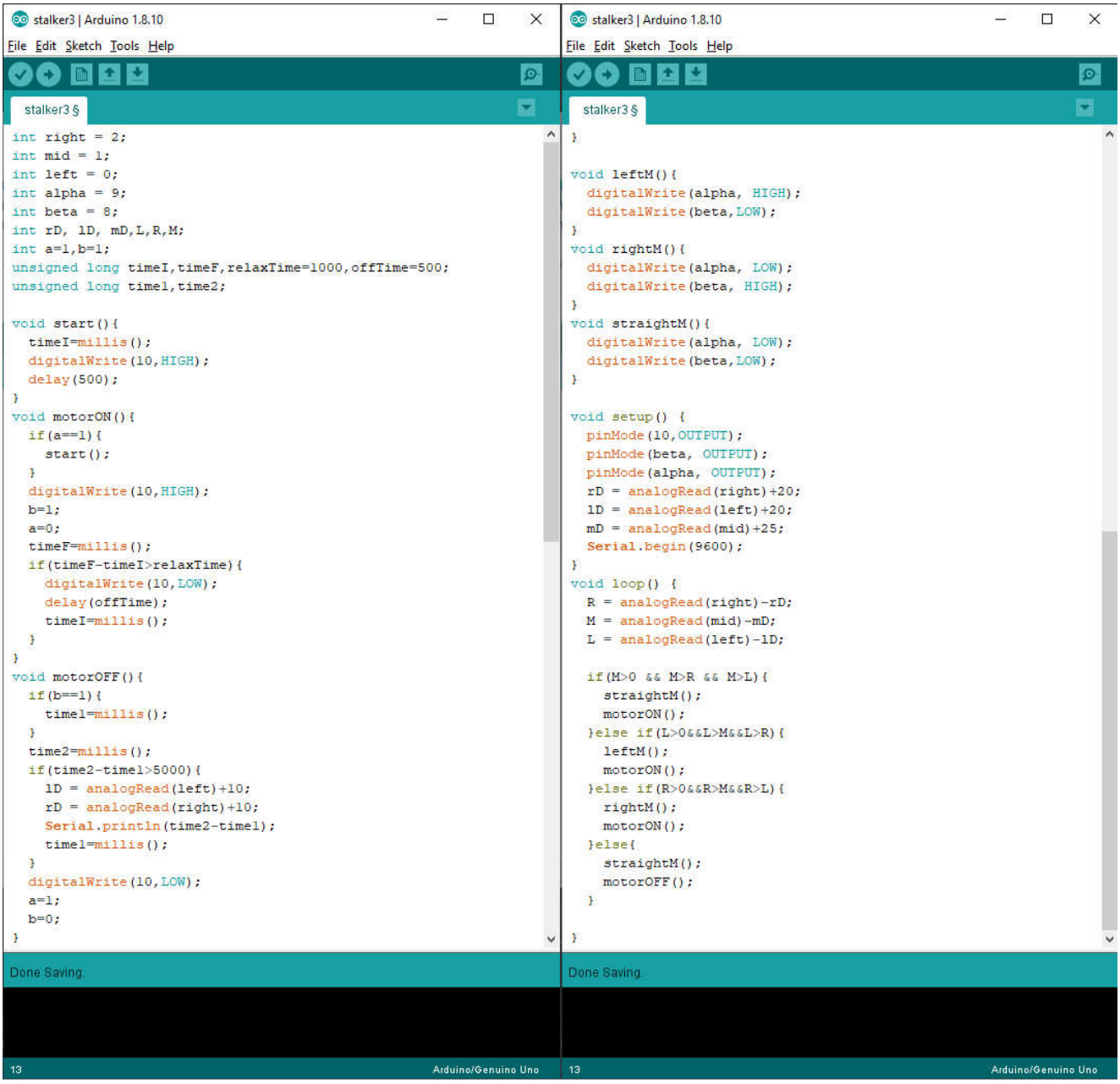
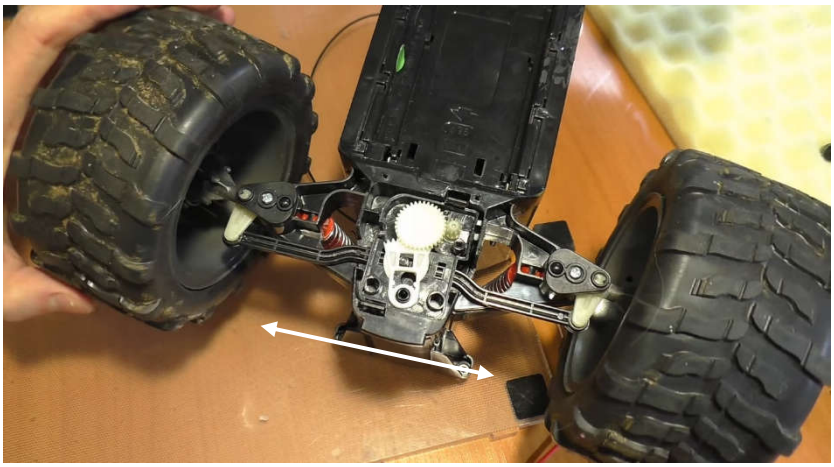


Fig.3. The sketch (code)

Basically, the Arduino constantly reads the Right, Left, and Middle photo-resistors (Fig. 3), and directs the car the changing the poles of ruling dc motor, using the H-bridge.



Unlike servo motors, ruling system of a toy cars based DC motor, so it must be under constant power in order to turn left and right.

Fig.4. Ruling mechanics

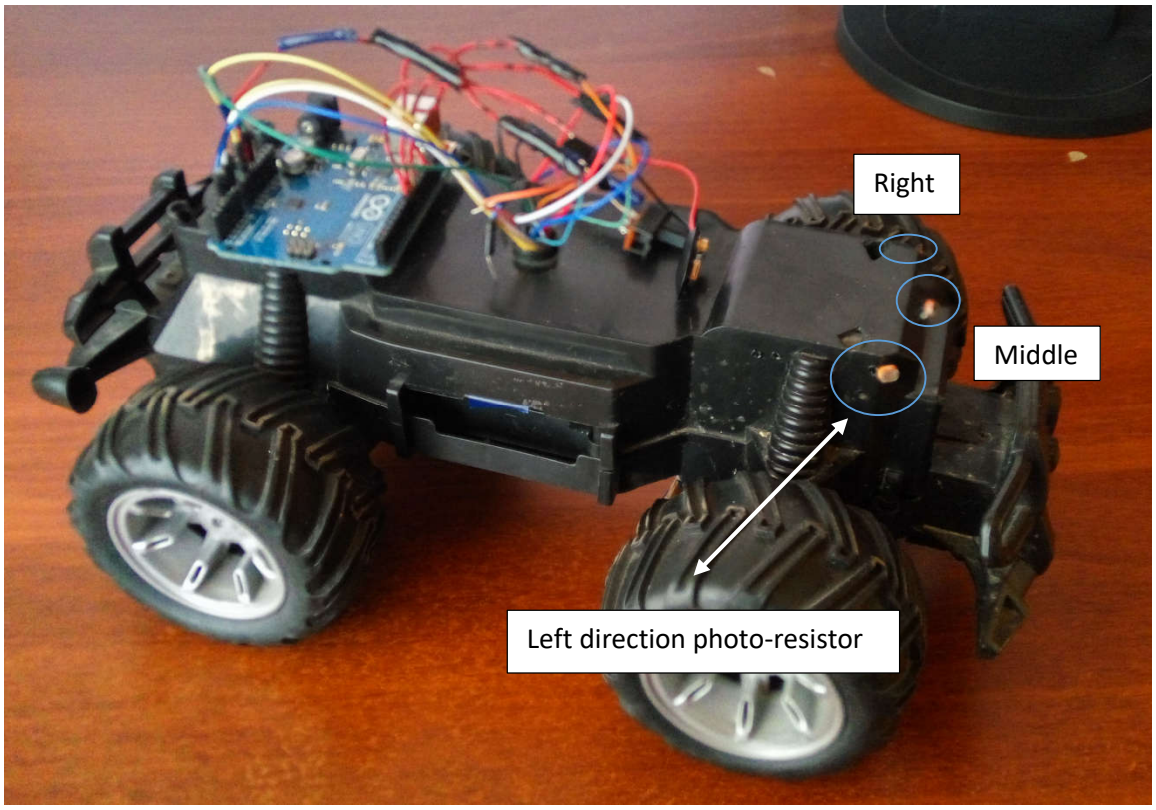


Fig.5. Finished project.
The light stalker

Environment illumination

In my project, I have been using photo-resistors as an indicator of source of light. On one hand, it is very cheap and available way compared to infrared indicator that are used all over the world. On other hand, that caused many issues. Firstly, photo-resistors by nature are identical, e.g. they react to the light differently, one may change for 5 units, other for 3 units, and so on. That was not so hard to solve in using Arduino. I just have defined for every pins (photo-resistors) different configurations.

However, the main problem was in the environment illuminations, other sources of light, lamps, the sun, etc. The solution for this problem was into making the car bit more intelligent, e.g. I made so that car constantly was adapting into the environment lightings, when idle state time was exceeding 5 sec.

```
void motorOFF(){
  if(b==1){
    time1=millis();
  }
  time2=millis();
  if(time2-time1>5000){
    lD = analogRead(left)+10;
    rD = analogRead(right)+10;
    Serial.println(time2-time1);
    time1=millis();
  }
  digitalWrite(10,LOW);
  a=1;
  b=0;
}

void setup() {
  pinMode(10,OUTPUT);
  pinMode(beta, OUTPUT);
  pinMode(alpha, OUTPUT);
  rD = analogRead(right)+20;
  lD = analogRead(left)+20;
  mD = analogRead(mid)+25;
  maxM = mD+80;
  maxR = rD+80;
  maxL = lD+80;
  Serial.begin(9600);
}
```

rD, lD, mD – are the barrier level, if the incoming source of light exceed these values, a robot start to react towards it. Thus, by constantly refreshing it was possible to make the robot readier into environment condition.

Moreover, by changing the configurations of particular variable, it made possible to use different Photo-resisters, as we are actually scaling them into needed state. As you see, middle max value barrier is different.

maxM,maxR,maxL we'll discuss later on

Driving motor issues

From a picture, it is seen (Fig.5) the machine is quite bulky. Hence, dragging power was accordingly. Indeed, that must not be a problem, but the toy car that I acquired didn't have built-in speed regulating functions, what is characteristic of toy cars. Due to these problems, bulkiness (inertia) and inability to regulate a speed, the acceleration and braking was not instant. Therefore, even though when a command "stop" was given to the device, it hadn't stop and may continue the movement further for 1-2 meters.

I made up two solutions in order to handle it.

1. Constantly turning on and off as the car moving, but this also caused an another sub-issue. It was not starting off since the car was intended to work under full load, and that amount of time when the driving motor was on, was simple enough to start the movement.

2. Starting the brake function beforehand. Certainly, that could be seen very obvious solution, but my detectors were photo-resistors, which compared to infra-red detectors, just physically can't calculate the distance before a source of light.

```
void loop() {
  R = analogRead(right)-rD;
  M = analogRead(mid)-mD;
  L = analogRead(left)-lD;
  Rx = analogRead(right);
  Mx = analogRead(mid);
  Lx = analogRead(left);
  if(M>0 && M>R && M>L && Mx<maxM){
    straightM();
    motorON();
  }else if(L>0 && L>M && L>R && Rx<maxR){
    leftM();
    motorON();
  }else if(R>0 && R>M && R>L && Lx<maxL){
    rightM();
    motorON();
  }else{
    straightM();
    motorOFF();
  }
```

Beforehand braking was accomplished, by manually checking the characteristics of every photo-resister individually (these are maxL, maxR, maxM variable in the void setup()) , at last It was found optimal value by exceeding which the car would want to start braking.

That variable was included into the ruling conditionals in the void loop(), along with directional conditions.

```
void start(){
  timeI=millis();
  digitalWrite(10,HIGH);
  delay(500);
}
void motorON(){
  if(a==1){
    start();
  }
  digitalWrite(10,HIGH);
  b=1;
  a=0;
  timeF=millis();
  if(timeF-timeI>relaxTime){
    digitalWrite(10,LOW);
    delay(offTime);
    timeI=millis();
  }
}
void motorOFF(){
  if(b==1){
    time1=millis();
  }
  time2=millis();
  if(time2-time1>5000){
    lD = analogRead(left)+10;
    rD = analogRead(right)+10;
    Serial.println(time2-time1);
    time1=millis();
  }
  digitalWrite(10,LOW);
  a=1;
  b=0;
}
```

Start() function ran once at the beginning of a motion, after it would run only if motorOFF() would be used. Implementation of this function solved the starting off problem.

As you in the body of MotorON(), I wrote an if condition block, that constantly analyzed whether when the movement state exceeded 1 sec, it would turn off the driving motor for a 0.5 sec, so it was not able to gain high speed, which aggravated the braking process.

motorOFF() ,as that may be obvious by its name, stops the motion of the car.