UNIVERSIDAD DEL TURABO

GURABO, PUERTO RICO

JOSÉ A. PÉREZ SCHOOL OF ENGINEERING

RC / Automated Grass Cutter

José A. Ortiz Rodríguez

S00818343

Capstone 2

Prof. Alcides Alvear

**Abstract**

Currently, mowing the lawn can be tiring and time consuming, there are also people who has disabilities and can’t do it. Therefore we are going to create an automated and remote control machine that can cut the grass for those people that cannot or do not want to mow their lawn. Our solution, is the creation of a fully electrical grass cutter which will be controlled by a mobile application or autonomously. The mobile application will communicate with the microcontroller via Bluetooth to control the lawn mower. The autonomous part will execute the task without the need of human intervention, it will cover the area following a pattern or randomly this will be done with a microcontroller. The power supply of this machine will be supplied by three 12 V batteries, which two will be connected in series "24 V" to power the motor of the cutter and the other battery of 12 V to feed the two motors which will move the machine and the microcontroller. It will have a charging station equipped with a weather sensor to determine if precipitation occurs to reschedule the cutting process. To conclude, the purpose of this project is to make people's lives more comfortable and simple at the time of mowing the grass, without almost any effort.

**Index**

1. Introduction …………………………………………………………………………. 4
   1. Problem Statement ………………………………………………………….. 5
   2. Mentors ……………………………………………………………………... 5
   3. Requirement
      1. Functional …………………………………………………………… 5
      2. Non-Functional ……………………………………………………… 5
   4. Diagrams
      1. Requirement Diagram ………………………………………………. 5
      2. Use Case Diagram ………………………………………………….. 7
   5. System description ………………………………………………………….. 8
   6. Methodology ………………………………………………………………... 8
2. System description
   1. System Architecture ………………………………………………………… 9
   2. Data Flow
      1. System design ………………………………………………………. 10
      2. Cyclomatic Complexity …………………………………………….. 11
   3. Components Architecture……………………………………………………. 12
   4. Estimated Material Cost …………………………………………………….. 14
   5. Implementation
      1. Circuit of the grass cutter …………………………………………… 15
      2. Development application …………………………………………… 16
      3. Autonomous mode ………………………………………………….. 17
3. Testing ………………………………………………………………………………. 19
4. Results ………………………………………………………………………………. 21
5. Code ………………………………………………………………………………… 22
6. Conclusion and Future work
   1. Conclusion …………………………………………………………………... 32
   2. Future work ………………………………………………………………….. 32
7. **Introduction**

Initially, today for many people it is difficult to cut the grasses either because they have a disability or unwilling to do so because it is very hard work.   In this work we are going to create a machine which solves this problem. The goal of this project is to be able to create a machine that we can control in a remote way, that is to say that the person when he goes to cut the grams does it of a simple form and without any effort. Also this machine will be operated by DC batteries, which reduces the gas emissions of the majority of conventional machines and does not affect our health directly. This will also have safety sensors to protect the person's well-being in case they pick up or use the machine in an inappropriate way.

* 1. **Problem statement**

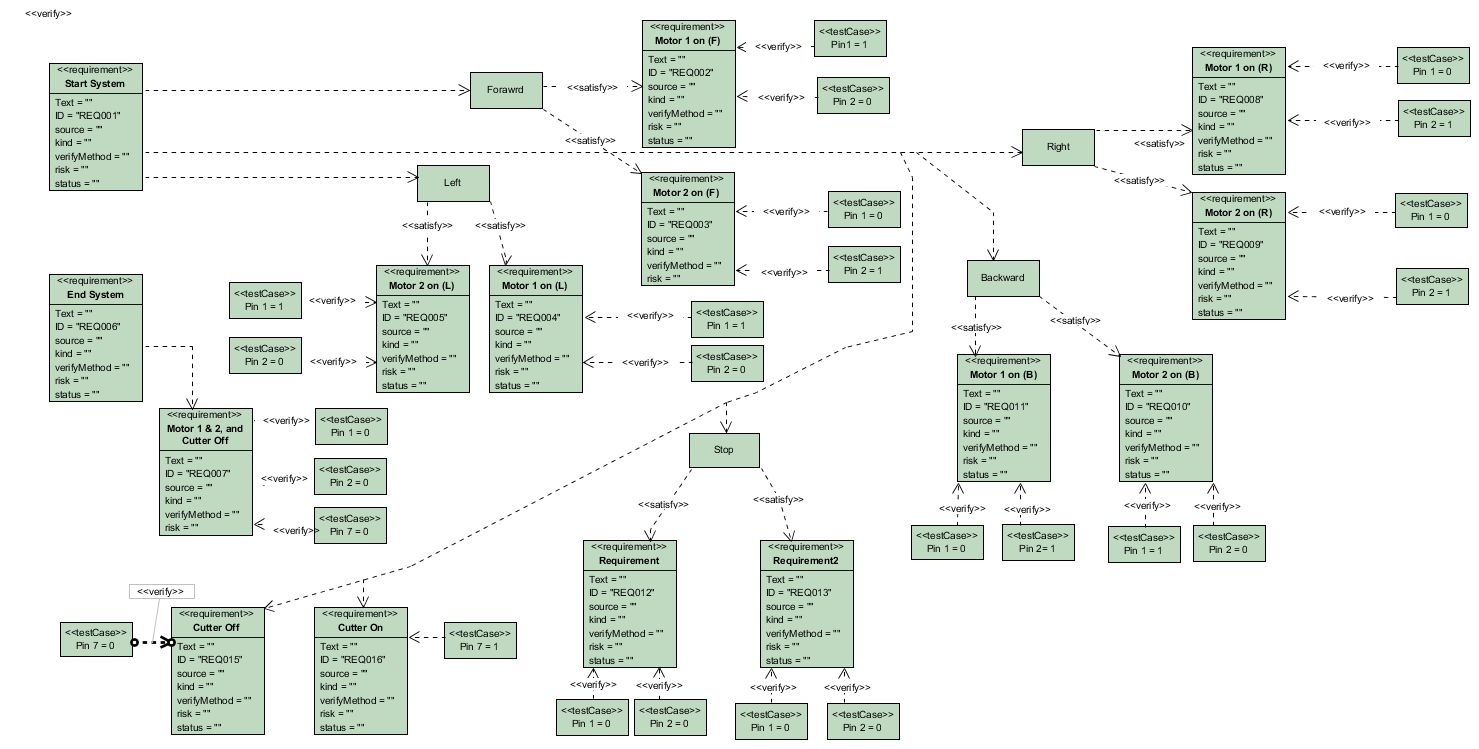
Mowing the lawn is an exhausting job, the people with disabilities are not able to do it, and pay for doing it is expensive, and reduce emissions and gasoline spills.

* 1. **Mentors**

The mentors of this project are:

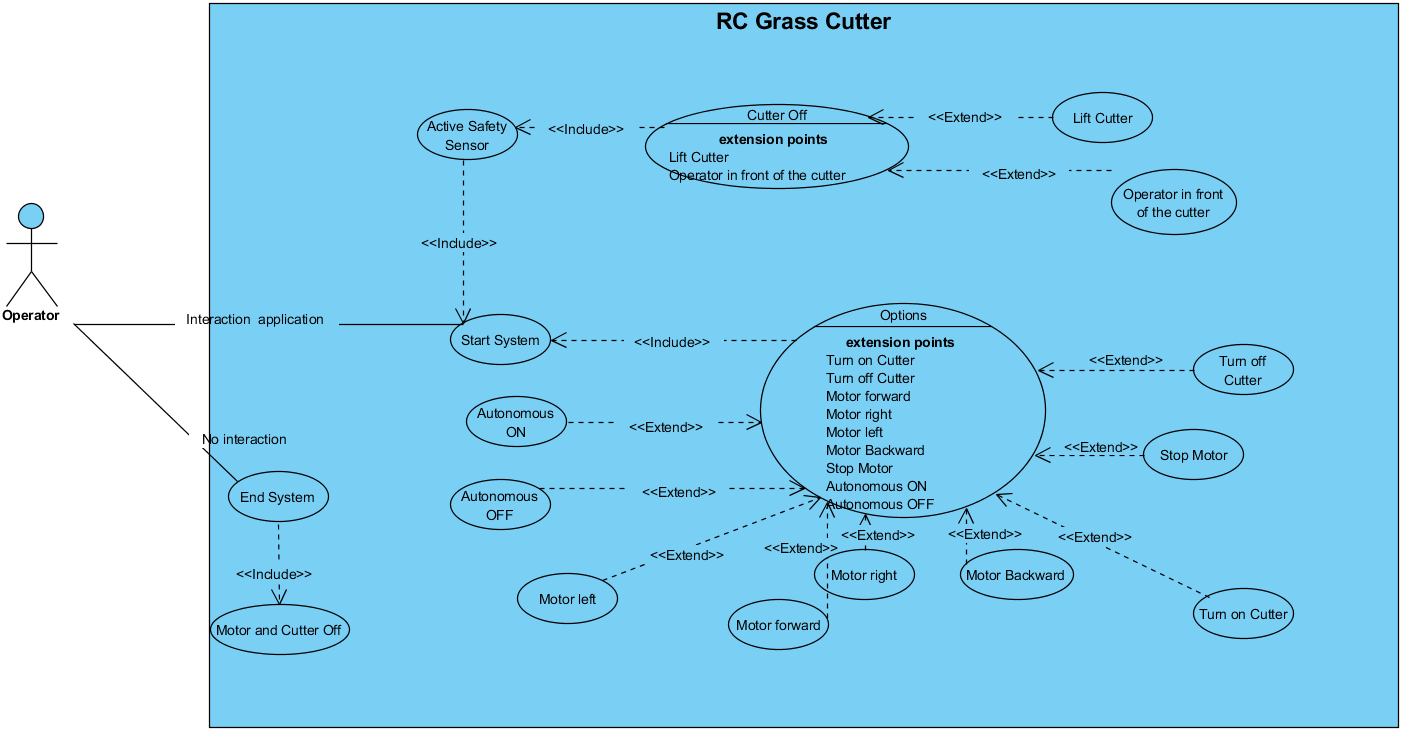
* Prof. Alcides Alvear
* Dr. Miguel Goenaga
* Prof. Diego A. Aponte
  1. **Requirement** 
     1. **Functional**
* A microcontroller to send signals to the H-Bridge to control the direction of the grass cutter.
* Communication with phone application.
* A mobile application to control de grass cutter.
* Cut the grass remotely via Bluetooth through a mobile application or autonomous mode
  + 1. **Non – Functional**
* Development the application in IOS
* Switch the grass cutter to autonomous mode
* Charging Station
* Power line communication
  1. **Diagrams** 
     1. **Requirement Diagram**

In the Figure 1 we can see a diagram of requirement which shows the different cases of how the grass cutter should act according to the action that we give through the mobile application, either to move to the front, back, left, right, stop, or to turn the cutter on or off.



*Figure 1: Requirement Diagram*

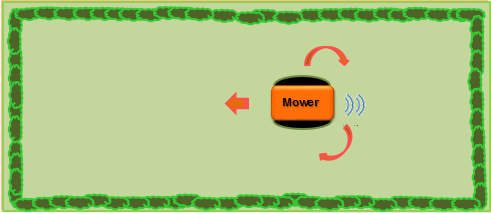
* + 1. **Use Case Diagram**

The figure 2 show how the operator interacts with the lawn mower. The operator interacts with the mobile application when the system starts, it has different options which it uses to control the lawn mower. If the lawn mower has an obstacle to the front or if it is lifted the safety sensors are activated and the lawn mower stops moving and cutting. Finally the system ends when the operator stops interacting with the lawn mower.

*Figure 2: Use Case Diagram of the RC/Automated Grass Cutter.*

* 1. **System description**

Our system have the ability to mow the lawn by controlling a mobile application via Bluetooth, this with almost no effort. In figure 3 we can see an image of what the interaction of the lawn mower would look like in any average yard.

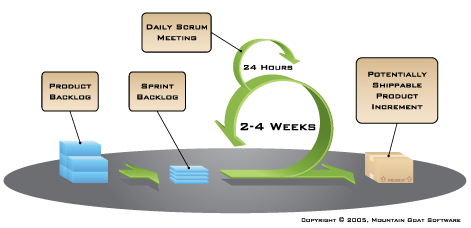




*Figure 3: System functionality Grass Cutter*

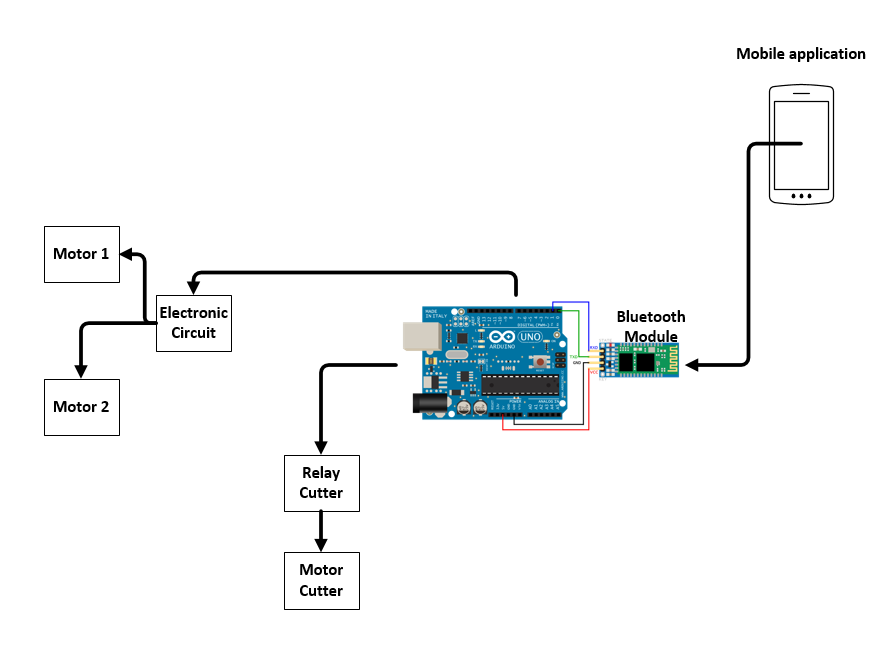
* 1. **Methodology**

The methodology used to develop part of this project it was the agile methodology Scrum. This allows in any time to realign the software with the objectives of the business or Project, since it can introduce functional or priority changes at the beginning of each new iteration without any problem. Also this methodology is flexibility to change the project and reduce the time to the market.



*Figure 4: Agile System Development Lifecycle*

1. **System Description**
   1. **System Architecture (Block Diagram)**

 The lawn mower system consists of a mobile application, Bluetooth module, Arduino (microcontroller), circuit, motors and a relay that activates or deactivates the motor of the mower. This works as follows; The user performs an action with the mobile application either turning the cutter on or off, moving the machine to the front, back, left, right or stop, this sends a variable to the Bluetooth module which communicates with the Arduino "microcontroller" then according to the action that carried out the user in the application we can move our machine through the motors through the circuit "H-bridge" or activate the motor of the cutter.

*Figure 5: Functionality Grass Cutter, Block Diagram*

* 1. ***Data flow*** 
     1. **System design**

In figure 6 we can see how the communication system works, when the mobile application is communicated with the Bluetooth module that is connected to the Arduino. First the mobile application "the remote control" is connected to the Arduino. Then the Arduino receives through the Bluetooth module the action that the user executed through the application "where the machine is going to move or activate the cutter", if the lawn mower has an obstacle in front or is lifted the security sensors detect it immediately and cancel the action that the user executes, but proceeds to the action that the user executed.



*Figure 6: Flow chart of the system functionality.*

* + 1. **Cyclomatic complexity**

The cyclomatic complexity is a software metric, used to indicate the complexity of a program. This is computed using the control flow graph of the program: the nodes of the graph correspond to indivisible groups of command might be executed immediately after the first command. In the figure 7 we can see the cyclomatic complexity of our flow graph (figure 6).



**Region**

**Node**

**Edge**

*Figure 7: Cyclomatic complexity diagram*

V (G) = numbers of regions = 4

V (G) = Edges – node + 2 = 16 -14 + 2 = 4

V (G) = Predicted nodes + 1 = 3 + 1 = 4

* Path 1: 1-2-3-4
* Path 2: 1-2-3-6-5
* Path 3: 1-2-3-6-7-8-3-4
* Path 4: 1-2-3-6-7-9-3-4
* Path 5: 1-2-3-6-7-10-3-4
* Path 6: 1-2-3-6-7-11-3-4
* Path 7: 1-2-3-6-7-12-3-4
* Path 8: 1-2-3-6-7-13-3-4
* Path 9: 1-2-3-6-7-14-3-4
* Path 10: 1-2-3-6-7-15-3-4
* Path 11: 1-2-3-6-7-16-3-4
  1. **Components Architecture**

**Resistors**

The resistor are used to generate the base current of the transistors.

**Power Supply**

The power in our system is supplied by two 12V 5A/h batteries and one 12V 7A/h used to power the Arduino and supply to the motors. 5A/h batteries are in series configuration to get 24V to power the motor that drives the cutting mechanism.

**Motors**

There are two 12V, 3Amp motor for propulsion and one 24V, 1.5Amp for cutting the grass.

**Diodes**

The diodes are used to protect the system. They eliminate the possibility of current entering currents through the emitters of the power transistors that are not in use.

**Transistors 2N2222A, Tip3055T y Tip42C**

We used these transistors to make an H-bridge configuration, to change the direction of the propulsion motors.

**Microcontroller**

An Arduino Uno is a board with a microcontroller Atmega 328, voltage regulators and a USB port connected to a USB-Serial adapter module that allows to program the microcontroller. This microcontroller is used to control the direction of the motor through its input and output port to move the grass cutter.

**Switch**

We used a double pole double throw switch to turn on or off the 12V and 24V power sources.

**Relay (SRD-05VDC-SL-C)**

This relay was used to power the 24V that drivers the cutting mechanism.

**Bluetooth HC-06**

This Bluetooth allows communication between the application and the microcontroller.

**HC-Sr04 Ultrasonic (random)**

This was used for the autonomous mode when the grass cutter moves randomly.

**Optical Flow V1.0 & ITG/MPU (patron)**

The Optical Flow is a camera which works like a mouse (x and y axis), we use it to read the distance that the cutter must travel. ITG / MPU this is an accelerometer and gyroscope of 6 degrees. This was used to turn the cutter to 180 degrees, after it stopped with the optical sensor thus creating a possible patron.

* 1. **Estimated material cost**

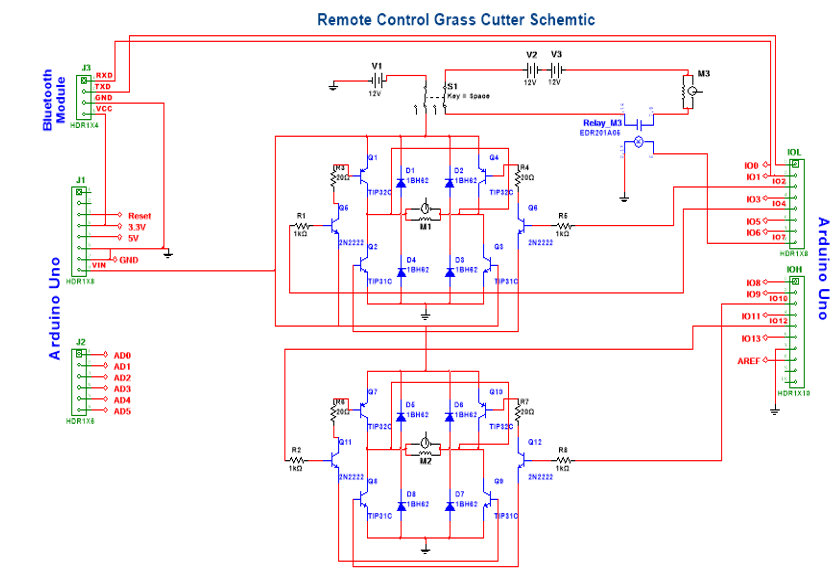
In the table 1 you can see an estimate of the costs of the materials.

**Table 1 Estimated Material Cost RC Grass Cutter**

|  |  |  |
| --- | --- | --- |
| **Estimated Material Cost** | | |
| Materials | Amount | Sub-Total |
| Arduino Uno | 1 | $ 5.00 |
| Diode | 8 | $ 4.00 |
| Transistor (2N2222a) | 4 | $ 4.00 |
| Relay (SRD-05VDC-SL-C) | 1 | $ 3.00 |
| Resistor | 12 | $ 1.20 |
| Transistor (Tip 31C) | 4 | $ 6.00 |
| Transistor (Tip 32C) | 4 | $ 6.00 |
| Batteries 12V | 3 | $ 75.00 |
| Motor 12V | 2 | $ 50.00 |
| Motor 24V | 1 | $ 20.00 |
| Bread Board | 1 | $ 2.50 |
| Switch | 1 | $ 2.00 |
| Bluetooth HC-06 | 1 | $ 2.75 |
| **Total** | | $ 181.45 |

* 1. **Implementation** 
     1. **Circuit of the grass cutter**

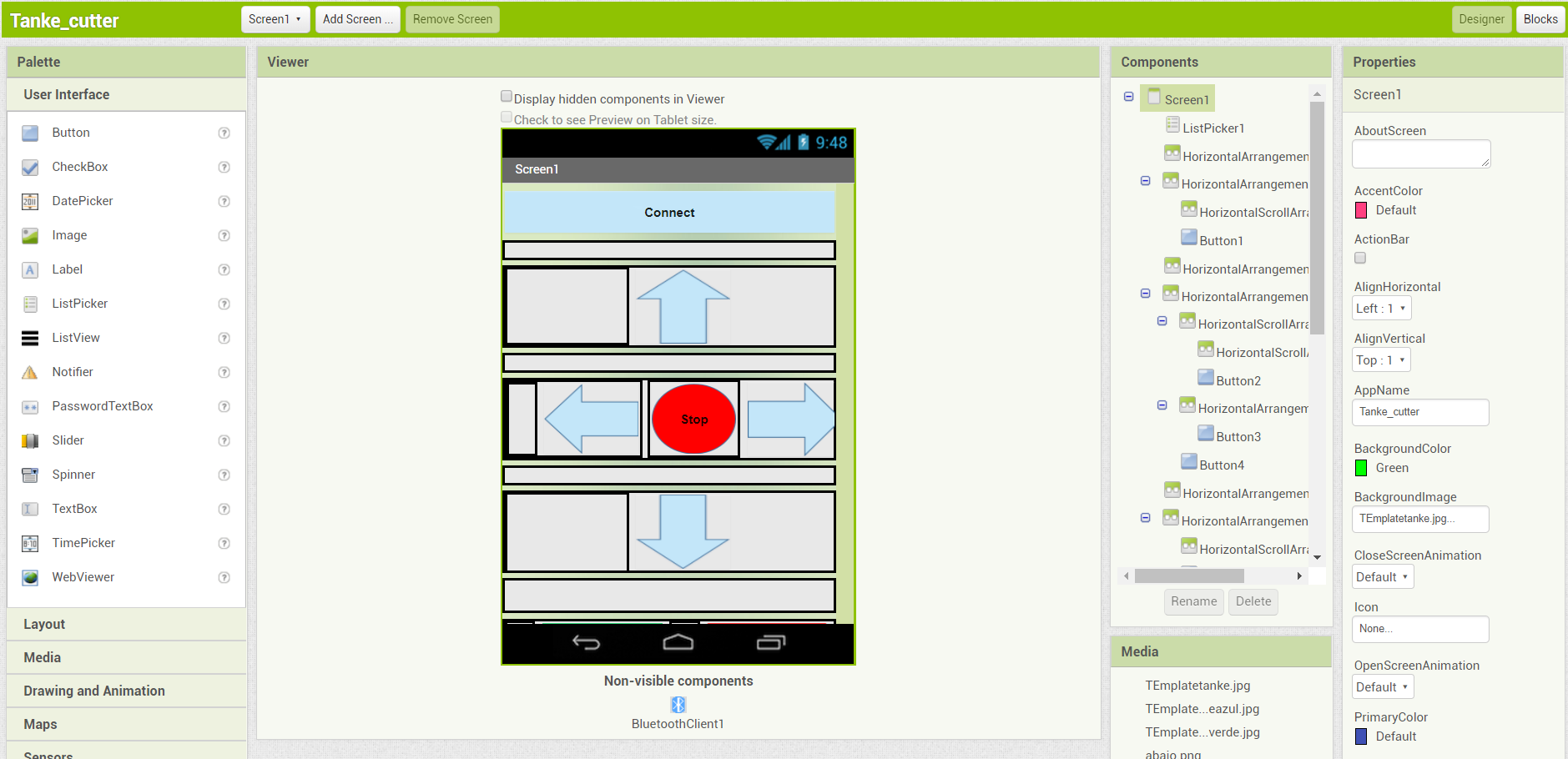
In Figure 8 we can see a complete diagram of the H-bridges and the Arduino connections. The function of the H-bridges is to give the motors direction for the front or back. The Arduino sends a pulse of 5V or 0V, which activates or deactivates the movement of the motor, which in our case we use to move the lawn mower.

****

*Figure 8: Schematic of the grass cutter “H-bridge”*

* + 1. **Development application**

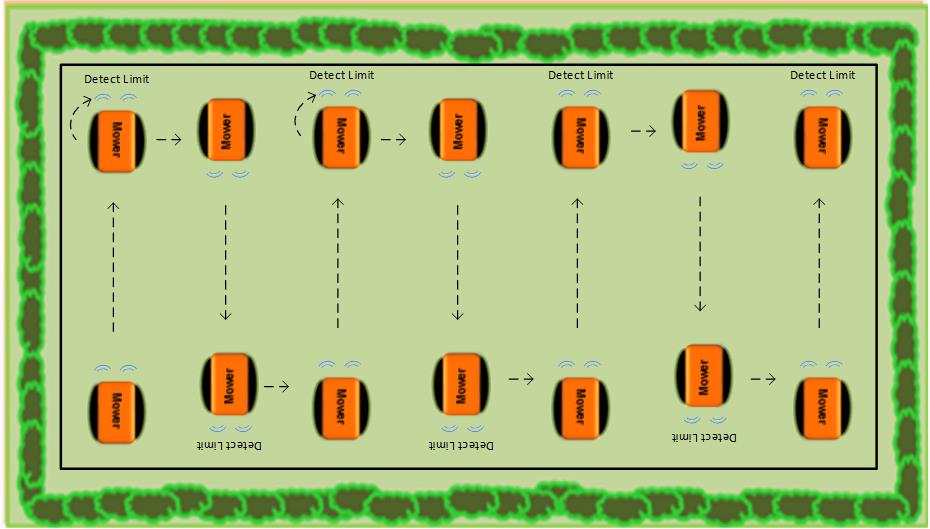
In Figure 9 we can see the page on which we are developing the application which is, MIT App Inventor. Basically this page is an open-source Web Development, in which we can develop applications in a simple way.



*Figure 9: Development application, MIT App Inventor*

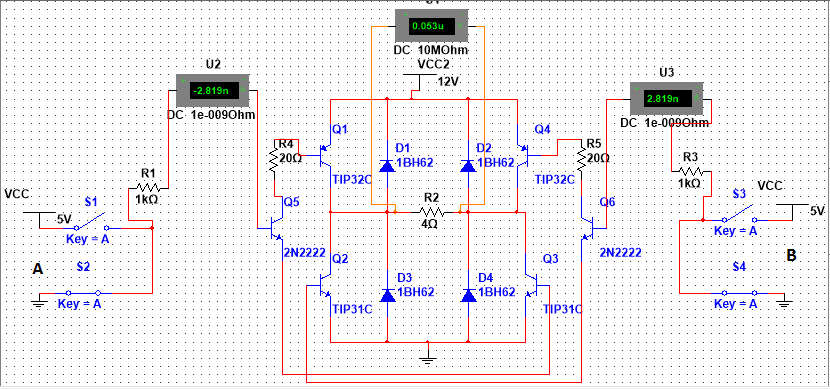
* + 1. **Autonomous mode**

In this mode the cutter must move independently, either with a pattern or randomly. In figure 10 we can see how would be the pattern of the cutter in this mode. *Note: this is a concept in this way, since there are many variables such as: the terrain and the texture of the grass, this proved its functionality and worked at 75%.*

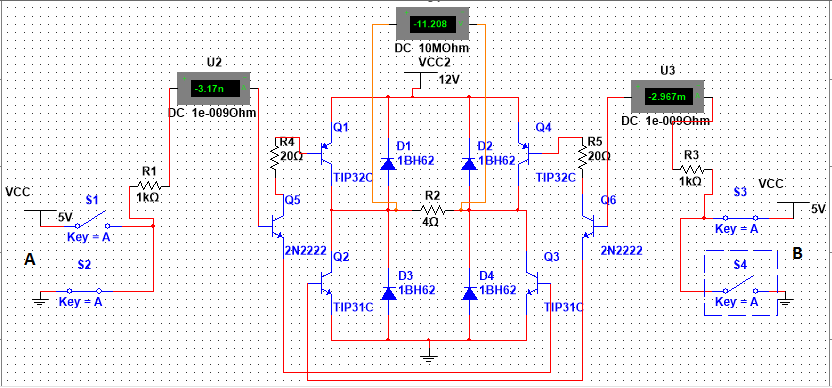
****

*Figure 10: Autonomous mode concept*

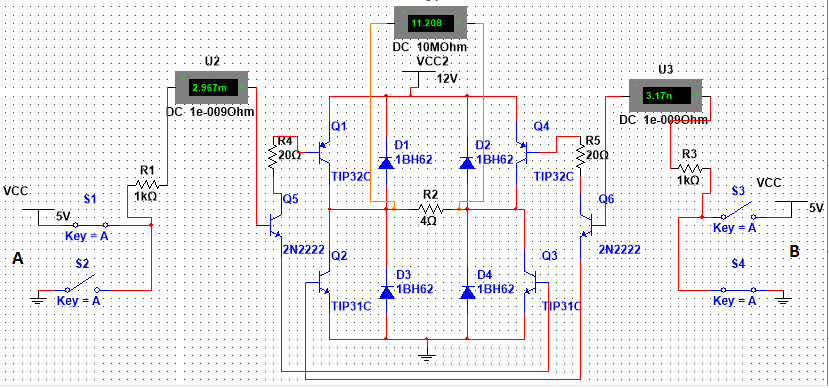
1. **Testing**



*Figure 11: H-Bridge, inputs A and B are both 0V.*



*Figure 12: H-Bridge, inputs A is 5V and B is 0V.*



*Figure 13: H-Bridge, inputs A is 0V and B is 5V.*

Figure 11 shows that when inputs are 0 the motor will receive 0V and it will not move. If inputs are 5V and 0V the motor will receive 11.08V (see figure 12) and will move forward while if inputs are 0V and 5V motor will receive -11.08V (see figure 13) and will move backwards. If both inputs are 5V it will short circuit the bridge and this state should not happen. All possible combinations and output are listed in table 2.

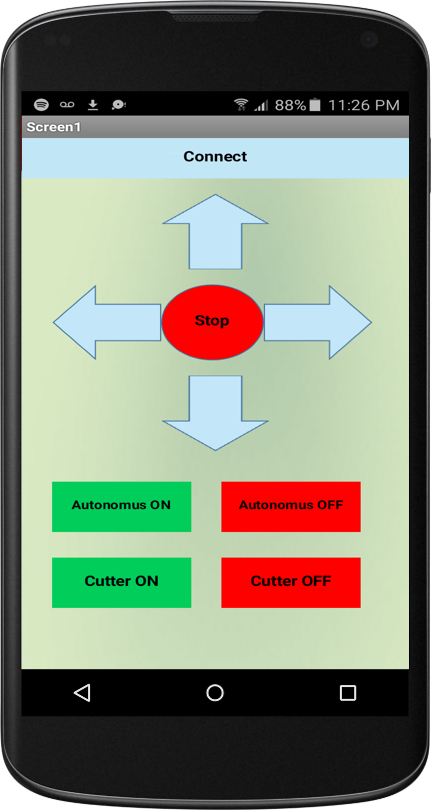
Table 2: H-bridge Truth table

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Output Voltage** |
| 0V | 0V | 0V |
| 5V | 0V | 11.08V |
| 0V | 5V | -11.08V |
| 5V | 5V | FORBIDDEN |

1. **Results**

**Android application**

Basically this is the application that we use to control the lawn mower and through this we connect to our machine via Bluetooth or autonomously, through the "Connect" button.

****

*Figure 14: Android application to control de grass cutter*

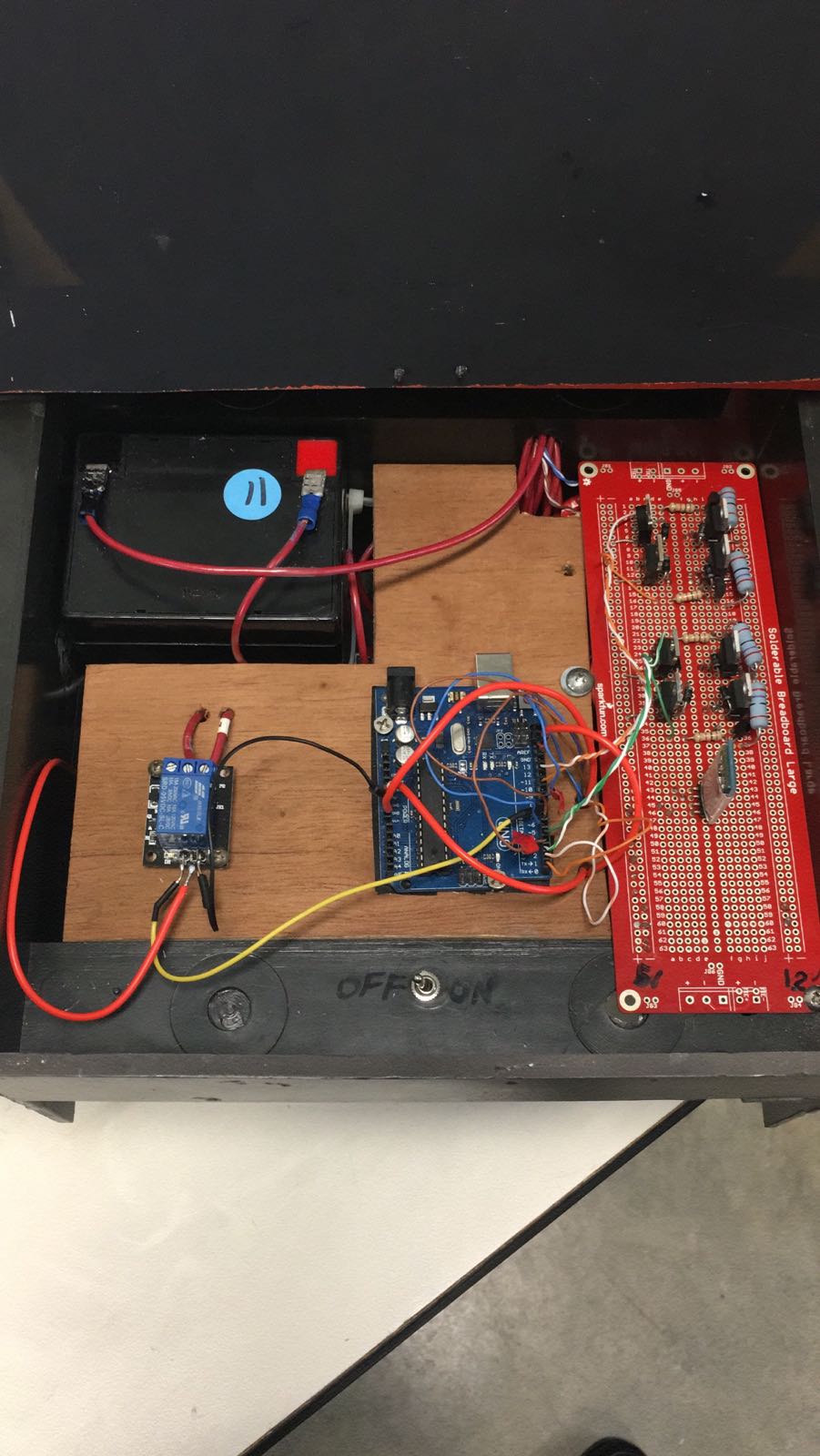
**Grass Cutter prototype**

In the figure 15 we can see what the lawn mower looks like.



*Figure 15: Grass cutter*

In the figure 16 and 17 we can see the connection in the lawn mower looks like.

* *

*Figure 16 & 17: Connection of the lawn mower.*

1. **Code**

*The code of this machine is development in Arduino IDE.*

//////////LIbrerias//////////////

#include <Wire.h>

#include <HMC5883L.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_HMC5883\_U.h>

HMC5883L compass;

// Create instance of digital compass

Adafruit\_HMC5883\_Unified mag = Adafruit\_HMC5883\_Unified(12345);

////////// Motors ///////////////////

const int motor1Pin1 = 12;

const int motor1Pin2 = 10;

const int motor2Pin1 = 4;

const int motor2Pin2 = 6;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*VARIABLE DIRECTION CONTROL\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

float initial\_heading;

float target\_heading;

float diff\_heading;

bool turn\_init;

bool target\_reached;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*Left & right\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

#define LEFT 0

#define RIGHT 1

int acc = 0;//Acumulacion de y

double Ang = 0;//Acumulacion de angulo

int serialA='S'; //inicia detenido

double x =0;

//////// Cutter /////////////////

const int pin7 = 7;

long distancia,duracion;

/////// Flow Sensor ///////////

#include "SPI.h"

// these pins may be different on different boards

// this is for mega

#define PIN\_SS 11

#define PIN\_MISO 50

#define PIN\_MOSI 51

#define PIN\_SCK 52

#define PIN\_MOUSECAM\_RESET 3

#define PIN\_MOUSECAM\_CS 2

#define ADNS3080\_PIXELS\_X 30

#define ADNS3080\_PIXELS\_Y 30

#define ADNS3080\_PRODUCT\_ID 0x00

#define ADNS3080\_REVISION\_ID 0x01

#define ADNS3080\_MOTION 0x02

#define ADNS3080\_DELTA\_X 0x03

#define ADNS3080\_DELTA\_Y 0x04

#define ADNS3080\_SQUAL 0x05

#define ADNS3080\_PIXEL\_SUM 0x06

#define ADNS3080\_MAXIMUM\_PIXEL 0x07

#define ADNS3080\_CONFIGURATION\_BITS 0x0a

#define ADNS3080\_EXTENDED\_CONFIG 0x0b

#define ADNS3080\_DATA\_OUT\_LOWER 0x0c

#define ADNS3080\_DATA\_OUT\_UPPER 0x0d

#define ADNS3080\_SHUTTER\_LOWER 0x0e

#define ADNS3080\_SHUTTER\_UPPER 0x0f

#define ADNS3080\_FRAME\_PERIOD\_LOWER 0x10

#define ADNS3080\_FRAME\_PERIOD\_UPPER 0x11

#define ADNS3080\_MOTION\_CLEAR 0x12

#define ADNS3080\_FRAME\_CAPTURE 0x13

#define ADNS3080\_SROM\_ENABLE 0x14

#define ADNS3080\_FRAME\_PERIOD\_MAX\_BOUND\_LOWER 0x19

#define ADNS3080\_FRAME\_PERIOD\_MAX\_BOUND\_UPPER 0x1a

#define ADNS3080\_FRAME\_PERIOD\_MIN\_BOUND\_LOWER 0x1b

#define ADNS3080\_FRAME\_PERIOD\_MIN\_BOUND\_UPPER 0x1c

#define ADNS3080\_SHUTTER\_MAX\_BOUND\_LOWER 0x1e

#define ADNS3080\_SHUTTER\_MAX\_BOUND\_UPPER 0x1e

#define ADNS3080\_SROM\_ID 0x1f

#define ADNS3080\_OBSERVATION 0x3d

#define ADNS3080\_INVERSE\_PRODUCT\_ID 0x3f

#define ADNS3080\_PIXEL\_BURST 0x40

#define ADNS3080\_MOTION\_BURST 0x50

#define ADNS3080\_SROM\_LOAD 0x60

#define ADNS3080\_PRODUCT\_ID\_VAL 0x17

void mousecam\_reset()

{

digitalWrite(PIN\_MOUSECAM\_RESET,HIGH);

delay(1); // reset pulse >10us

digitalWrite(PIN\_MOUSECAM\_RESET,LOW);

delay(35); // 35ms from reset to functional

}

int mousecam\_init()

{

pinMode(PIN\_MOUSECAM\_RESET,OUTPUT);

pinMode(PIN\_MOUSECAM\_CS,OUTPUT);

digitalWrite(PIN\_MOUSECAM\_CS,HIGH);

mousecam\_reset();

int pid = mousecam\_read\_reg(ADNS3080\_PRODUCT\_ID);

if(pid != ADNS3080\_PRODUCT\_ID\_VAL)

return -1;

// turn on sensitive mode

mousecam\_write\_reg(ADNS3080\_CONFIGURATION\_BITS, 0x19);

return 0;

}

void mousecam\_write\_reg(int reg, int val)

{

digitalWrite(PIN\_MOUSECAM\_CS, LOW);

SPI.transfer(reg | 0x80);

SPI.transfer(val);

digitalWrite(PIN\_MOUSECAM\_CS,HIGH);

delayMicroseconds(50);

}

int mousecam\_read\_reg(int reg)

{

digitalWrite(PIN\_MOUSECAM\_CS, LOW);

SPI.transfer(reg);

delayMicroseconds(75);

int ret = SPI.transfer(0xff);

digitalWrite(PIN\_MOUSECAM\_CS,HIGH);

delayMicroseconds(1);

return ret;

}

struct MD

{

byte motion;

char dx, dy;

byte squal;

word shutter;

byte max\_pix;

};

void mousecam\_read\_motion(struct MD \*p)

{

digitalWrite(PIN\_MOUSECAM\_CS, LOW);

SPI.transfer(ADNS3080\_MOTION\_BURST);

delayMicroseconds(75);

p->motion = SPI.transfer(0xff);

p->dx = SPI.transfer(0xff);

p->dy = SPI.transfer(0xff);

p->squal = SPI.transfer(0xff);

p->shutter = SPI.transfer(0xff)<<8;

p->shutter |= SPI.transfer(0xff);

p->max\_pix = SPI.transfer(0xff);

digitalWrite(PIN\_MOUSECAM\_CS,HIGH);

delayMicroseconds(5);

}

// pdata must point to an array of size ADNS3080\_PIXELS\_X x ADNS3080\_PIXELS\_Y

// you must call mousecam\_reset() after this if you want to go back to normal operation

int mousecam\_frame\_capture(byte \*pdata)

{

mousecam\_write\_reg(ADNS3080\_FRAME\_CAPTURE,0x83);

digitalWrite(PIN\_MOUSECAM\_CS, LOW);

SPI.transfer(ADNS3080\_PIXEL\_BURST);

delayMicroseconds(50);

int pix;

byte started = 0;

int count;

int timeout = 0;

int ret = 0;

for(count = 0; count < ADNS3080\_PIXELS\_X \* ADNS3080\_PIXELS\_Y; )

{

pix = SPI.transfer(0xff);

delayMicroseconds(10);

if(started==0)

{

if(pix&0x40)

started = 1;

else

{

timeout++;

if(timeout==100)

{

ret = -1;

break;

}

}

}

if(started==1)

{

pdata[count++] = (pix & 0x3f)<<2; // scale to normal grayscale byte range

}

}

digitalWrite(PIN\_MOUSECAM\_CS,HIGH);

delayMicroseconds(14);

return ret;

}

void setup() {

pinMode(motor1Pin1, OUTPUT);

pinMode(motor1Pin2, OUTPUT);

pinMode(motor2Pin1, OUTPUT);

pinMode(motor2Pin2, OUTPUT);

pinMode(pin7,OUTPUT);

pinMode(PIN\_SS,OUTPUT);

pinMode(PIN\_MISO,INPUT);

pinMode(PIN\_MOSI,OUTPUT);

pinMode(PIN\_SCK,OUTPUT);

SPI.begin();

SPI.setClockDivider(SPI\_CLOCK\_DIV32);

SPI.setDataMode(SPI\_MODE3);

SPI.setBitOrder(MSBFIRST);

Serial.begin(9600);

///////Compass////////

// Initialize Initialize HMC5883L "Sensor"

// Initialise the sensor

if(!mag.begin())

{

/\* There was a problem detecting the HMC5883 ... check your connections \*/

Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");

while(1);

}

// Initialise direction variables

turn\_init = false;

target\_reached = false;

//////////////Flow Sensor/////////////////

if(mousecam\_init()==-1)

{

Serial.println("Mouse cam failed to init");

while(1);

}

}

char asciiart(int k)

{

static char foo[] = "WX86\*3I>!;~:,`. ";

return foo[k>>4];

}

byte frame[ADNS3080\_PIXELS\_X \* ADNS3080\_PIXELS\_Y];

void loop() {

//////////////// Sensor "Optical Flow" ///////////////////////////

#if 0

// if enabled this section grabs frames and outputs them as ascii art

if(mousecam\_frame\_capture(frame)==0)

{

int i,j,k;

for(i=0, k=0; i<ADNS3080\_PIXELS\_Y; i++)

{

for(j=0; j<ADNS3080\_PIXELS\_X; j++, k++)

{

Serial.print(asciiart(frame[k]));

Serial.print(' ');

}

Serial.println();

}

}

Serial.println();

delay(250);

#else

// if enabled this section produces a bar graph of the surface quality that can be used to focus the camera

// also drawn is the average pixel value 0-63 and the shutter speed and the motion dx,dy.

int val = mousecam\_read\_reg(ADNS3080\_PIXEL\_SUM);

MD md;

mousecam\_read\_motion(&md);

for(int i=0; i<md.squal/4; i++)

Serial.print('\*');

Serial.print(' ');

Serial.print((val\*100)/351);

Serial.print(' ');

Serial.print(md.shutter); Serial.print(" (");

Serial.print((int)md.dx); Serial.print(','); //AQUI VARIABLES X Y

Serial.print((int)md.dy); Serial.println(')');

acc = acc + md.dy;

// Serial.println(md.max\_pix);

delay(100);

#endif

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Angle move "HMC5883" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// Read which button is pressed

int key\_value = 138;

/\* // KEY1

if (key\_value > 130 && key\_value < 140) {

turn(90, RIGHT);

}

// KEY2

if (key\_value > 570 && key\_value < 580) {

turn(90, LEFT);

}\*/

//////////////////////// Option Grass Cutter ///////////////////////

if (Serial.available() > 0) {serialA = Serial.read();}

////// Foward ////////

if (serialA == 'F'){

Forward();

}

////// Left ////////

if (serialA == 'L'){

Left();

}

////// Right ////////

if (serialA == 'R'){

Right();

}

////// Reverse ////////

if (serialA == 'B'){

Reverse();

}

////// Stop ////////

if (serialA == 'S'){

Stop();

}

////// Cutter On ////////

if (serialA == 'C'){

digitalWrite(pin7,HIGH);

}

////// Cutter Off ////////

if (serialA == 'c'){

digitalWrite(pin7,LOW);

}

////// Autonomous On ////////

if (serialA == 'A'){

if (acc > -500){ // Aqui cambiamos la distancia

Forward();

}

else {

Stop();

delay(2000);

if (key\_value > 130 && key\_value < 140) {

turn(90, RIGHT);

acc = 0 ;

}

}

}

////// Autonomous Off ////////

if (serialA == 'a'){

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, LOW);

digitalWrite(motor2Pin2, LOW);

digitalWrite(pin7,LOW);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funciones de las direcciones cortadora \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

}

void Forward()

{

digitalWrite(motor1Pin1, HIGH);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, LOW);

digitalWrite(motor2Pin2, HIGH);

}

void Left()

{

digitalWrite(motor1Pin1, HIGH);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, HIGH);

digitalWrite(motor2Pin2, LOW);

}

void Right()

{

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, HIGH);

digitalWrite(motor2Pin1, LOW);

digitalWrite(motor2Pin2, HIGH);

}

void Reverse()

{

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, HIGH);

digitalWrite(motor2Pin1, HIGH);

digitalWrite(motor2Pin2, LOW);

}

void Stop()

{

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, LOW);

digitalWrite(motor2Pin2, LOW);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funciones para movimiento angulo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// Turn of a given angle

void turn(float angle, bool turn\_direction) {

// Get initial heading

if (!turn\_init) {

Serial.println("Starting turn ...");

initial\_heading = get\_heading();

if (turn\_direction) {

target\_heading = initial\_heading + angle;

}

else {

target\_heading = initial\_heading - angle;

}

Serial.print("Initial heading: ");

Serial.println(initial\_heading);

Serial.print("Target heading: ");

Serial.println(target\_heading);

turn\_init = true;

target\_reached = false;

diff\_heading = 0;

}

float heading = get\_heading();

if (turn\_direction) {

diff\_heading = target\_heading - heading;

}

else {

diff\_heading = heading - target\_heading;

}

while (diff\_heading > 0 && !target\_reached) {

// Move & stop

if (turn\_direction) {

//right\_angle(255);

Right();

}

else {

//left\_angle(255);

Left();

}

delay(450);

//right\_angle(0);

Right();

delay(450);

// Measure heading again

float heading = get\_heading();

if (turn\_direction) {

diff\_heading = target\_heading - heading;

}

else {

diff\_heading = heading - target\_heading;

}

Serial.print("Difference heading (degrees): "); Serial.println(diff\_heading);

if (diff\_heading < 0 && !target\_reached) {

target\_reached = true;

turn\_init = false;

diff\_heading = 0;

}

return 'A';

}

// Stop

Serial.println("Stopping turn ...");

}

// Get heading from the compass

float get\_heading() {

// Get a new sensor event

sensors\_event\_t event;

mag.getEvent(&event);

// Calculate heading based on magnetic intensity in X & Y axis

float heading = atan2(event.magnetic.y, event.magnetic.x);

// Correct for when signs are reversed.

if(heading < 0)

heading += 2\*PI;

// Check for wrap due to addition of declination.

if(heading > 2\*PI)

heading -= 2\*PI;

// Convert radians to degrees for readability.

float headingDegrees = heading \* 180/M\_PI;

return headingDegrees;

}

1. **Conclusion and Future work**
   1. **Conclusion**

In this project we could find a small problem which we tried to find a solution. The problem with this job was that it is difficult for some people or they cannot cut the grass, our solution was to create a lawn mower which they could handle through a mobile application remotely and without any effort. During the development of this project we endorsed our knowledge learned in class and learned new skills which allowed us to reason in an orderly way to carry out this work and find a correct solution. The part a little more complicated of this project was the part of the bridges H since this heated us, but quickly we found a solution. The part of the mobile application was a bit laborious since it was something new, but it could be carried out in a simple and comfortable way. Our goal for this work is to be able to continue perfecting this project to be able to make it autonomous and to be able to make it commercial.

* 1. **Future work**

The future work of this Project is the implementation of autonomous capability, safety features capabilities, change propulsion motors to 24V, create a charging station, improve the mobile device application and develop an IOS application.