# PROYECTO FINAL DE INGENIERÍA

# DESARROLLO DE UN ROBOT AUTOMÓVIL MULTISENSORIAL PARA LA OBTENCIÓN DE PARÁMETROS DEL MEDIO AMBIENTE CIRCUNDANTE

**Perez Rodil, Juan Carlos– LU1074944**

Ingeniería Electromecánica

**Ritacco Fernández, Franco– LU1094437**

Ingeniería Electromecánica

Tutor:
**Zambrano, Daniel, UADE**

Co-Tutor:
-

Colaborador/es:
-

**2019**


**UNIVERSIDAD ARGENTINA DE LA EMPRESA**

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

# Agradecimientos

***Agradecemos a nuestro tutor, Daniel Zambrano por su ayuda a lo largo del trabajo, y por su método de enseñanza; siempre apostando a la realización de proyectos, a la investigación y al desarrollo de nuevas capacidades.***

***A nuestras familias, amigos y parejas que mediante su apoyo nos dieron la energía necesaria para encarar esta finalización de la carrera con mucha fuerza y siempre con el objetivo en mente de ser llamados "Ingenieros".***

## Resumen

La finalidad del proyecto es la realización de un dispositivo móvil multisensorial, que permita la obtención de parámetros del medio ambiente. Se dará un contexto histórico acerca de las razones para el uso de estos dispositivos, seguida de una introducción al proyecto, para luego proceder a la realización del mismo.

Se trabajará, en principio, sobre la selección de los componentes necesarios para la realización del proyecto y sus respectivos costos, y luego se abordarán las formas de implementación, conexiones electrónicas, programación, interfaz de usuario y diseño mecánico de las distintas partes que lo componen.

Al final, se mostrará el prototipo realizado, las oportunidades vistas de escalabilidad del proyecto y las conclusiones, que abordaran las complicaciones y ventajas del mismo.

## Abstract

The purpose of this project, is the development of a multisensory mobile device, which will allow to obtain parameters of the circulating environment. A brief of the historical context about the reason for the uses of these devices will be shown, followed by an introduction to the project; to then proceed to the implementation itself.

First, the selection of components necessary for the project development and their respective costs will be addressed. Afterwards, the ways of implementation as programming, electrical connections, user interface and mechanical design, will be explained in detail.

In the end, the created prototype will be shown, as well as the improvement opportunities and the conclusions, which will aboard the complications and advantages of the project itself.

# Índice

# 1. Introducción

El objetivo del proyecto es el de desarrollar y diseñar un robot automóvil económico, que, mediante la utilización de sensores y un mando a control remoto para movilizarlo, pueda dar información acerca de las propiedades del medio ambiente que lo rodea. El proyecto consiste es realizar un diseño de detalle, que contenga todos los elementos para la construcción futura del robot y su implementación en otros proyectos.

Este robot alcanzara lugares para los cuales el ser humano tiene dificultades de acceso, y ahorrara la utilización de muchos artefactos de medición diferentes, teniendo todos ellos en un solo producto y con la información disponible de una manera cómoda y eficiente.

Esto tendrá relevancia en cuestiones de seguridad industrial, para realizar mantenimientos preventivos ahorrándose el uso de distintos elementos de medición, centralizándolos todo en uno, y agregándole el factor de movilidad que permite al usuario mantener cierta distancia de la zona, y en estudios de ergonomía y condiciones laborales. Permitirá realizar inspecciones antes de cada proceso para prevenir accidentes debido a factores tales como la temperatura, la humedad, fugas de gases, etc. También podrá ser utilizado de manera didáctica, para prácticas en Universidades o Escuelas.

Elegimos empezar por la parte electrónica; todo lo relacionado a las conexiones eléctricas, alimentación, selección del mejor controlador para el proyecto, conexión (usando simuladores de conexión como Fritzing) y elección de los sensores más eficientes y útiles a la hora de recolectar datos ambientales. Luego, seguiremos por el desarrollo de las partes del armazón del carro. Se hará la elección del material y de las piezas necesarias para el proyecto, utilizando un software de CAD para su diseño y ensamble.

El controlador será programado dependiendo el elegido; se armará el código con todas las funciones de movimiento del carro y manejo de datos necesarias. Por último, se elegirá el protocolo de comunicación con el usuario y el modo de mando. Es importante la realización de un análisis de los tipos de interfaz de usuario más requeridas y la forma más cómoda de visualización de datos.

De esta forma, se terminará dando un diseño completo y en detalle para la construcción del producto.

# 2. Antecedentes

Un **robot** es una máquina programada y controlada para manipular objetos, moverse y realizar trabajos al mismo momento que interacciona con su entorno. Tiene como principal objetivo sustituir al ser humano en tareas repetitivas, difíciles, desagradables e incluso peligrosas de una forma más segura, precisa y rápida.

Cuando las máquinas además de comunicar y controlar comenzaron a dirigir operaciones, cumplir órdenes e incluso aprender, se desarrolló una nueva ciencia: la Cibernética, palabra que deriva del vocablo griego *Kybernetes*, que significa "timonel". Fundada en la década de 1940 por el matemático norteamericano Norbert Wiener, es la ciencia que estudia la comunicación entre el hombre y la máquina, y entre las propias máquinas.

El área de conocimiento en la que se enmarco la Robótica fue la Automática, la cual se definió por la Real Academia de las Ciencias como la disciplina que se ocupa de los métodos y procedimientos cuya finalidad es la sustitución del operador humano por un operador artificial en la ejecución de una tarea física y mental, previamente programada. A partir de esta definición, en la Automática se diferenciaron dos componentes claros:

- Una ***Unidad de Control*** que gobierna las acciones a realizar. Este gobierno debe cumplir ciertos criterios u objetivos del control como la estabilización ante perturbaciones, o la evolución temporal y el comportamiento dinámico óptimo respecto a determinados parámetros de calidad. Los avances en el campo de la inteligencia artificial permiten dotar a estas unidades de aspectos más avanzados como la toma de decisiones o el aprendizaje.

- Un ***Actuador*** que realiza las acciones programadas bajo la supervisión de la unidad de control. Estos dispositivos pueden ir desde los casos más elementales, como accionadores hidráulicos, neumáticos o electromecánicos hasta máquinas más complejas como manipuladores, máquinas-herramientas y, quizás los autómatas por excelencia, los robots.

La coordinación entre ambos componentes mediante el intercambio de información es lo que permitió que las tareas solicitadas se realizaran de manera correcta. Puesto que es posible definir la Informática como la ciencia que estudia el tratamiento de la información, es evidente que existe una relación clara entre Automática e Informática.

Luego, el control por realimentación, el desarrollo de herramientas especializadas y la división del trabajo en tareas más pequeñas fueron ingredientes esenciales en la automatización de las fábricas en el siglo XVIII. En la década de 1890 el científico Nikola Tesla, entre muchos otros dispositivos, ya construía vehículos controlados a distancia por radio.

La aparición del microcontrolador revoluciono la robótica, fue inventado en Texas Instruments en la década de 1970, alrededor del mismo tiempo que el primer microprocesador estaba siendo inventado en Intel. Los primeros microcontroladores eran simplemente microprocesadores con una función de memoria, como la memoria RAM y ROM. Más tarde, los microcontroladores se desarrollaron en una amplia gama de dispositivos diseñados para aplicaciones de sistemas embebidos específicos en dispositivos tales como automóviles, teléfonos móviles y electrodomésticos.

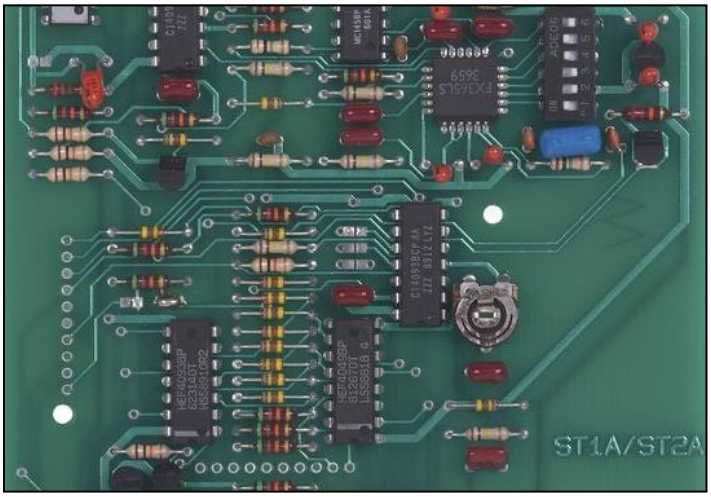



Figura 2.1: Primer Microcontrolador - Texas Instrument

Con el avance de la robótica, uno de los campos de aplicación más desarrollados fue la **robótica móvil**. La misma consiste en robots capaces de operar en condiciones poco favorables, riesgosas y sobre terrenos no aptos. Entre ellos, robots planetarios, robots en agricultura, robot en operaciones de búsqueda y rescate, robots militares, etc.

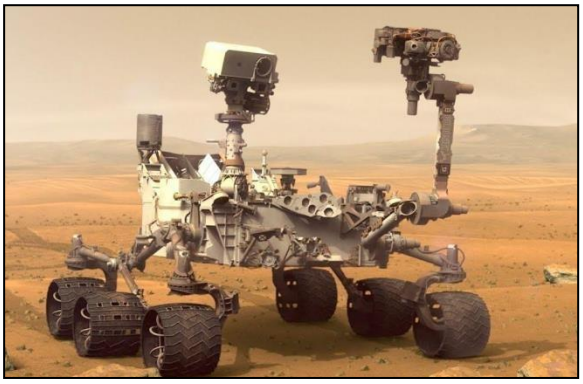


Figura 2.2:Robot *Curiosity* Marte

Figura 2.3: Robot Agrícola para Gestión de viñedos

En la última década la utilización de robots para operaciones de búsqueda y rescate aumento considerablemente lo que marca una tendencia a nivel global a la hora de realizar actividades riesgosas para el humano.

En septiembre del 2017, México sufrió un terremoto de alta magnitud y la Universidad Panamericana envió a la Ciudad de México, tras el suceso, a una brigada equipada con *IxnamikiOlinki*, (“máquina buscadora de personas”), en náhuatl; un robot especializado en tareas de rescate de personas en zonas de desastres. El mismo es capaz de llegar a lugares

difíciles de acceder o demasiado arriesgados para los seres humanos, además puede detectar personas y porta consigo instrumentos de auxilio.

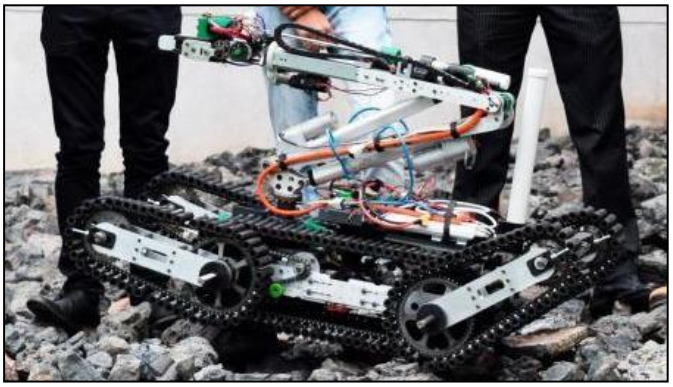


Figura 2.4: *IxnamikiOlinki*

El mismo está dotado con las siguientes herramientas:

- Una cámara térmica cuyo objetivo es la búsqueda de supervivientes en las zonas más afectadas.
- Un detector para medir los niveles de C02
- Un brazo mecánico capaz de realizar trabajo pesado o delicado, según requiera la situación, y con el que puede portar aparatos de comunicación, medicamentos y agua, de ser necesario.
- Integra conexión WI-FI o Ethernet.
- Un sistema de energía fotovoltaica para utilizarse por la noche, con una capacidad de 6 horas de autonomía.
- Componentes que le suministran de la fuerza de arranque y autonomía necesaria para desplazarse.

Sin lugar a dudas, la tecnología ha sido pieza clave para reducir los daños en pérdidas de vidas, y de gran ayuda para aquellos que realizaron tareas de salvamento.

Otro caso que tuvo como protagonista a la robótica fue el incendio de *NotreDame*. El 15 de abril de 2019 el incendio supuso el esfuerzo heroico de docenas de personas. Sin embargo, parte del mérito lo tiene *Colossus*, un robot de extinción de incendios por control remoto que pesa 500 kilos y maneja la Brigada de Bomberos de París.

Figura 2.5: Robot *Colossus* - Incendio de *Notre Dame* 15 de Abril, 2019.

***Colossus*** está diseñado para soportar condiciones extremas, operar una manguera contra incendios, transportar herramientas pesadas en un entorno hostil o extraer a personas heridas. Puede manejarse a casi 300 metros de distancia, y su diseño modular hace que se pueda configurar para diferentes trabajos.

Tiene aproximadamente 1,6 metros de largo, 0,75 metros de ancho y algo menos de 0,8 metros de alto. Es a prueba de agua, resistente al fuego, lo impulsan dos motores eléctricos que funcionan con seis baterías de iones de litio y tiene un sistema de frenos electromagnéticos.

Podemos observar que esta tecnología está siendo utilizada de manera recurrente, y provee un alivio a las necesidades de seguridad, control y movilidad que restringen al ser humano a la hora de la realización de procesos.

# 3. Desarrollo

El desarrollo consta de principalmente de dos partes, distribuidas en cada una de las secciones del proyecto: por un lado, la selección de los mejores componentes para las funciones propuestas junto con la justificación de su elección. Por el otro lado, la implementación de estos componentes para el diseño final.

## 3.1. Selección de Componentes

### 3.1.1. Microcontrolador

Un microcontrolador es un tipo de circuito integrado programable, que sirve para controlar elementos tanto de entrada como de salida, mediante los sistemas que lo componen. Funciona como una pequeña computadora, que incluye un procesador y memoria para guardar los programas que nosotros carguemos.


Figura 3.1.1.1: Microcontrolador (microchip).

Hay que hacer una diferencia entre lo que es un chip microcontrolador y lo que es una placa microcontroladora. Lo dispuesto en la Figura 3.1.1.1 es un chip microcontrolador que es un circuito integrado que puede ser programado, responder a cualquier tipo de instrucción en su memoria, y posee una unidad central de procesamiento (CPU), periféricos y, como fue dicho anteriormente, memoria.

La placa microcontroladora contiene un chip microcontrolador, pero esta anexada por pines de E/S digitales y/o analógicos. Estos le dan la capacidad al microcontrolador de conectarse a otros circuitos y así poder armar sistemas de control complejos. También suelen poseer un tipo de alimentación por USB y Jack de alimentación, y ya vienen con la capacidad de ser utilizables por cualquier IDE (Entorno de desarrollo integrado) para facilitar el uso inexperto.

Esto tiene muchas similitudes y suele ser sinónimo de una placa computadora o **SBC** (Single Board Computer).

Hay dos marcas de SBC hoy en día que son las más utilizadas en el mercado para proyectos de control de bajo coste: **Arduino y Raspberry Pi**. Ambos son de código abierto, es decir, pueden ser programados libremente permitiendo la colaboración abierta de la comunidad en proyectos sobre los mismos

Aquí vemos un resumen de algunas de las diferencias entre estas SBC (ver Tabla 1).

TABLA I: Diferencias técnicas entre Arduino y Raspberry Pi

| Criterio\ SBC | **Arduino** | **Raspberry** |
|---|---|---|
| **Procesamiento** | ATMEGA 328/2560 | Microprocesador 256/512 MB RAM |
| **Operaciones** | No tiene sistema operativo. | Sistema operativo propio: Raspbian |
| **Conexión a Internet** | Necesita un escudo (shield) para acceso a internet. | Puerto Ethernet |
| **Periféricos** | USB, Pines GPIO | USB, HDMI, RCA, Audio 3,5mm, Pines GPIO |
| **Velocidad** | 16 MHz | 700 MHz |
| **Utilización** | Fácil Control Electrónico | Fácil Control Informático |

Al ver estas diferencias, vemos que el Raspberry Pi tiene mayor potencialidad de usos que el controlador básico Arduino. La Raspberry Pi tiene mayor velocidad de procesamiento, más cantidad de periféricos y más cantidad de conexiones en la placa básica. Tiene todas las características más representativas de un **microordenador** y esa es la razón principal por la cual NO lo elegimos como nuestra opción para este proyecto.

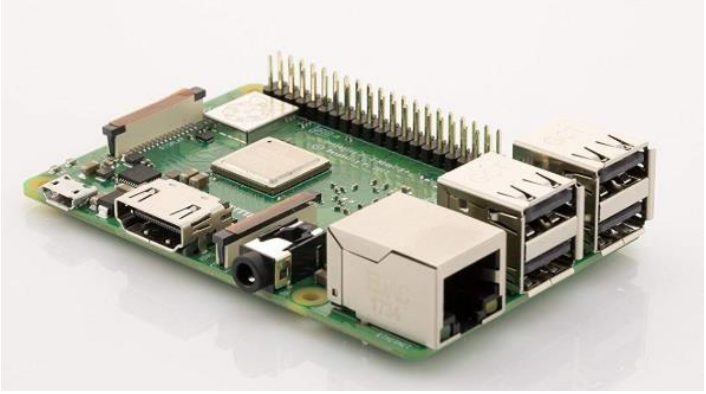


Figura 3.1.1.2: Raspberry PI 3 Modelo B+

Las razones principales para NO elegir el Raspberry Pi en el proyecto son las siguientes:

1. Necesidad de utilización de Sistema Operativo Raspbian para la programación.
2. El diseño tiende a ser el de un ordenador en sí, eliminando versatilidad en su uso.
3. Dificultades en su uso para circuitos electrónicos sencillos y mayor lentitud para su ejecución inicial, complicando su uso en proyectos de Electrónica, pero potenciándose en proyectos de Informática.
4. Requiere capacitación mucho mayor para su utilización.

**Arduino** nos ofrece una simplicidad suficiente para nuestro proyecto, con una mayor versatilidad, facilidad de uso y programación; y especializándose su uso para la electrónica, con informática de fácil acceso y entendimiento. Además, existen una gran cantidad de sensores muy fácilmente programables por este artefacto, y muchas librerías de programación para estos. Por estas razones, es el tipo de dispositivo elegido para este proyecto.

Los modelos de Arduino son variados, pero nos concentramos en la comparación entre 3 de ellos. Todos ellos operan a 5V y trabajan entre 7V -12V de entrada. Los modelos son:

1. **Arduino Nano**: la placa más compacta de la marca, basada en el procesador ATMega328P. Tiene 14 pines de entrada/salida digital y 6 analógicas, cristal de 16 MHZ, botón de reset, terminales que permiten un tipo de conexión ICSP y, el detalle

más característico; posee una conexión Mini-USB. Es muy parecido al Arduino UNO, con diferente conexión USB, con pines *header* y sin Jack de alimentación.

2. **Arduino UNO**: El modelo más clásico y documentado de Arduino. Es la placa más utilizada de todas, y posee las mismas características que Arduino Nano: 14 pines digitales, 6 analógicos, cristal de 16 MHZ, conexión ICSP y botón de reseteo; salvo por que los pines NO son *header*, posee un puerto USB y Jack de alimentación.
3. **Arduino Mega 2560**: basada en el microcontrolador ATMega2560. Posee 54 pines digitales, 16 analógicas, 4 UARTs, un cristal de 16 MHz, conexión USB, Jack para alimentación DC, conector ICSP y también, el botón de resteo.


Figura 3.1.1.3: Placa Arduino UNO

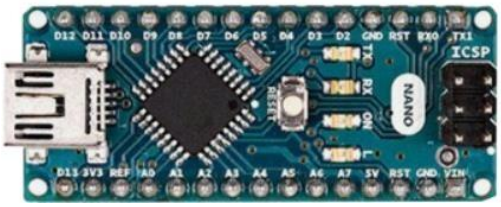



Figura 3.1.1.4: Placa Arduino NANO


Figura 3.1.1.5: Placa Arduino MEGA

Para nuestro proyecto, consideramos que el **Arduino MEGA 2560**es la mejor elección principalmente porque al ser un proyecto con mucha cantidad de conexiones, actuadores (motores) y sensores, se necesitan muchos pines para su realización. Este caso es ideal ya que

el Arduino MEGA posee 54 pines digitales y 16 analógicos, una cantidad significativamente mayor que su competencia. No existe una gran diferencia en el costo del producto respecto de los otros modelos, siendo así una opción eficiente y de bajo costo.

### 3.1.2. Sensores

Los **sensores** son los elementos de medición, en este caso electrónicos, que detectan la magnitud de parámetros físicos, y esto lo traduce para que el sistema pueda interpretarlo a través de una señal eléctrica. Los datos enviados por un sensor pueden ser **digitales** (Ejemplo: sensor de gas detecta gas en el ambiente) o pueden ser **analógicos**; impulsos con cierto dato numérico dado por el voltaje de la señal (Ejemplo: sensor de temperatura detecta 23`C).

Cabe recalcar que, para este proyecto, su función es tener un uso de **control básico** en industrias; decidimos concentrarnos en la utilización de sensores que puedan ser útiles en una planta para relevar datos e información sobre el ambiente que rodea cierta zona. Estos sensores NO reemplazarían a los de niveles industriales instalados en máquinas o en zonas de alto riesgo, que poseen una mayor seguridad y precisión. Este podrá ser utilizado para tareas de mantenimiento preventivos, inspección de sótanos o depósitos u otros espacios confinados. En base a esos datos se pueden tomar decisiones sobre cómo proceder a dicha área. Pero es importante determinar cuáles son los parámetros importantes que medir en cada industria.

Por eso es indispensable que la construcción de este proyecto sea fácil de entender y este bien especificada: así se pueden hacer modificaciones e implementar nuevos sensores dependiendo de la industria o situación en la que está siendo usado. Por medio de la programación se pueden setear diferentes **valores de control**, para distintas zonas o circunstancias en las que los parámetros se quieran medir. Pero también es necesario dejar la puerta abierta a la adición de nuevo hardware que pueda aumentar las funcionalidades para cada industria específica.

Los tipos de sensores que planteamos colocar en el proyecto son los que muestran los parámetros básicos de ambiente y algún posible riesgo sencillo de medir. Los más comunes y que analizaremos en este proyecto son:

- Temperatura
- Humedad Relativa

- Gases
- Presión Barométrica
- Luminosidad

Para poder categorizar y seleccionar el mejor sensor de cada tipo para el proyecto vamos a utilizar los siguientes medibles y criterios de selección, que aplicaran para varias de las categorías de sensores:

- **Rango de Medición**: es el rango de valores dentro de los cuales el sensor tiene las propiedades para dar un resultado. **Ejemplo**: 0 C - 60 C.
- **Precisión**: responde a la dispersión del conjunto de valores obtenidos de diferentes medidas realizadas de maneras reiteradas de cierta magnitud. El parámetro estadístico que lo define es el "desvió estándar". Ver Fig.7.1.1.6. **Ejemplo**: +- 2 C.

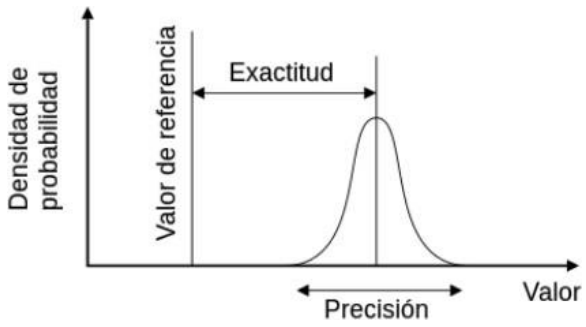



Figura 3.1.1.6: Muestra Grafica de la Precisión en una Distribución Normal

- **Resolución:** Mínima medida de cambio en una magnitud detectada por el sensor. **Ejemplo:** 0.01 C.
- **Tiempo de Respuesta:** Tiempo que tarda un cambio en la entrada, en verse reflejado en el valor de salida. **Ejemplo**: 1 segundo.
- **Costo**: El costo del sensor será tomado en consideración.
- **Otros**: Beneficios adicionales que pueda aportar un sensor respecto a otro o problemas que puede llegar a tener. Sera especificada si es Negativa, Neutral o Positiva con los colores.

Por supuesto, estos no son los únicos medibles que existen, pero son fáciles de ver y de interpretar, y los principales a la hora del planteo de cualquier proyecto. Esta información se encuentra mucho más detallada en el **Datasheet** de cada instrumento de medición, adjuntos en el ANEXO. Cabe destacar, que estos medibles no serán utilizados en todas las categorías de sensores, sino en las que corresponda utilizar estos datos. Otros medibles pueden ser útiles en otros casos particulares.

La Proximidad no la tomaremos como relevante, ya que el sensor tomara un parámetro ambiental promedio. No estamos pidiendo que el sensor tome la información a la salida de una cañería, o en una máquina, sino en el ambiente general y para esto, no es necesaria ningún tipo de análisis de proximidad.

#### 3.1.2.1. Sensores Ambientales: Temperatura, Humedad Relativa y Presión.

Para la elección de este grupo de sensores, se debieron tomar ciertos supuestos. En principio, investigamos las condiciones en las que debe encontrarse la locación en donde se utilice proyecto. Para esto se supuso una locación, que cumpla gran parte de las características para las cuales este proyecto fue diseñado; el **taller mecánico**. Dentro de las industrias se tienen talleres de mantenimiento o manufactura donde se realizan tareas de mecanizado, soldadura, etc. Estos lugares deben tener condiciones óptimas para el grupo de trabajo que este en él.

La **temperatura** en un taller mecánico, basados según las disposiciones mínimas de seguridad y salud dispuestas en el trabajo (Anexo II del Real Decreto 496/1997, España) la temperatura debe situarse entre los **14ºC y los 25ºC**. Tomamos como rango medible en territorio argentino de **-10ºC y 60ºC,** por supuesto, dando cierta holgura en los extremos por algún tipo de falla. La precisión no es tan importante en este caso; para parámetros ambientales, 1ºC nos parece correcto, ya que no influirá significativamente en lo que percibirá el individuo si se acercase a la zona. Los otros parámetros, siempre serán lo mínimo posible y elegiremos el componente en función de cómo se desempeña en estos otros aspectos.

Los sensores elegidos para analizar fueron los siguientes y los datos extraídos y comparados fueron mostrados en la Tabla II.

TABLA II: Análisis de Sensores de Temperatura

| Temperatura | Rango de Mediciones | Precision | Resolucion | Costo | Otros |
|---|---|---|---|---|---|
| Taller Mecanico | -5ºC - 60ºC (ideal 14ºC-25ºC) | ± 1ºC - ± 2ºC | Minimo posible | Minimos posible | "-" |
| LM35 | 2C - 150ºC | ±0,5° C | N/A | USD 1,75 | Analogico. Requiere voltajes negativos para dar Temperaturas Negativas |
| TMP36 | -55ºC - 150ºC | ±2° C | N/A | USD 2,25 | Analogico. Maneja Temperaturas Negativas con Voltajes positivos |
| DHT11 | 0ºC - 150ºC | ±2° C | 1° C | USD 2,50 | Digital. Es ademas Sensor de Humedad. |
| DHT22 | -40º C- 80º C | ±0.5° C | 0.1° C | USD 7,50 | Digital. Es ademas Sensor de Humedad. |
| BME280 | -40º C- 85º C | ±1° C | 0.01° C | USD 9,82 | Digital. Es ademas Sensor de Humedad y Presion. Muy rapida respuesta. |
| SHT3x | -40º C- 123.8º C | ±0.2° C | 0.015° C | USD 15,20 | Digital. Es ademas Sensor de Humedad. Muy rapida respuesta. |

Dentro de estos sensores, vemos que el **DHT22** cumple características interesantes. Cumple con los requisitos de rango de medición y precisión. Incluso posee un costo competitivo para sus funcionalidades; ofreciendo no solo un sensor de temperatura, sino que también de **Humedad Relativa;** otro de los parámetros que planteamos serian medidos por el robot en este proyecto. Al ser digital también provee protección contra el ruido.

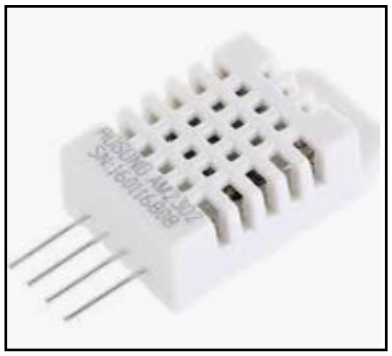


Figura 3.1.2.1.1: Modulo Sensor DHT22 de Temperatura y Humedad

Comparándolo con el **SHT3x**, que posee características superiores en cada uno de los aspectos (especialmente en la precisión), vemos una diferencia substancial de precio, no justificada para este proyecto y descartando así esta opción.

La desventaja del DHT22, es el tiempo **de las lecturas** (2 segundos). La toma de datos se hace muy lenta. Además, muchas reseñas muestran una baja fiabilidad en este sensor debido a problemas en su fabricación.

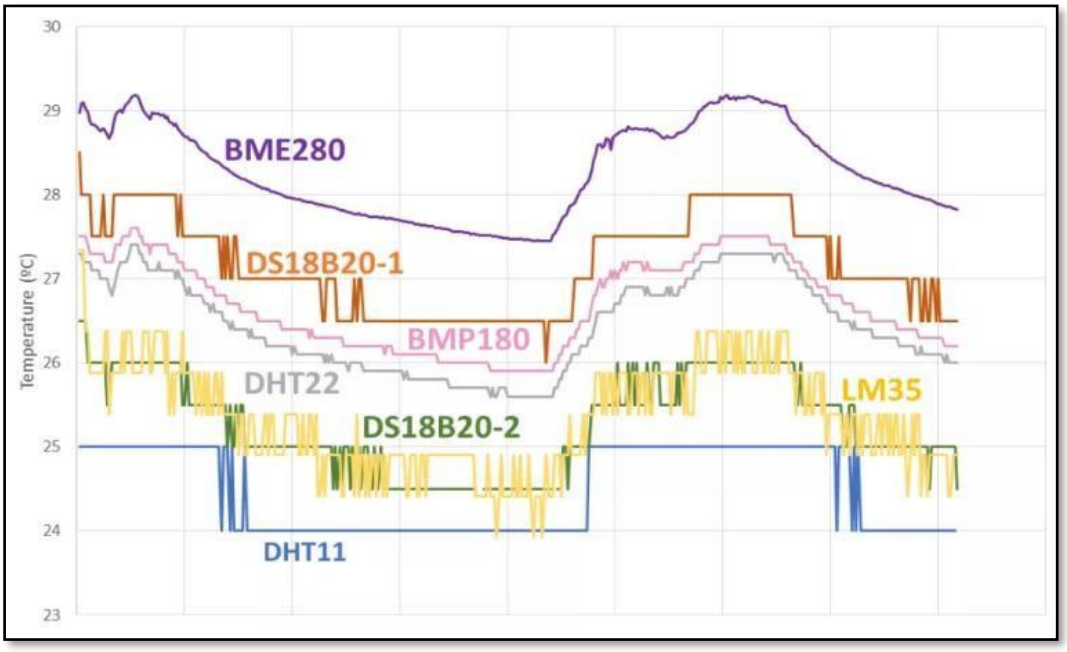



Figura 3.1.2.1.2: Mediciones de múltiples sensores en un ambiente a lo largo del tiempo.

Para poder ver las diferencias en detalle, teniendo en cuenta que gran parte de los sensores superaban los medibles dispuestos en la tabla, ubicamos un gráfico que testea diferentes sensores de temperatura a lo largo del tiempo y sus mediciones (ver Figura 3.1.2.1.2). Este grafico no posee la temperatura medida por un termómetro calibrado, pero la forma de las curvas nos da una idea de cómo se comportan.

Podemos ver que el **BME280** midió más alto que el resto, ya que el sensor tiende a calentarse, como el fabricante aclara en el Datasheet (ver Anexo). Sin embargo, es el sensor que dio la temperatura más estable, sin grandes oscilaciones entre las lecturas. Esto se debe a la buena resolución que detecta cambios de hasta 0.01º C.

Se puede ver en otros sensores, la gran cantidad de oscilaciones que poseen respecto al BME280, particularmente los sensores como el LM35. Se ven casos como el DHT11 que no puede detectar cambios pequeños en las temperaturas debido a su resolución de 1º C.

Vemos que la opción del **BME280**, salvo por el pequeño corrimiento de temperatura por calentamiento interno, se perfila como el elegido. Este único defecto, es fácilmente solucionable al programar la recepción de datos.

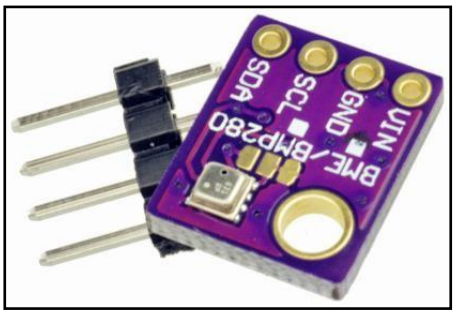



Figura 3.1.2.1.3: Sensor BME280 de Temperatura, Humedad Relativa y Presión

**Ventajas:**

- Rango de temperaturas suficiente.
- Precisión sobresaliente.
- Ofrece también medición de **Humedad Relativa y Presión Barométrica**.
- Precio competitivo, teniendo en cuenta sus funcionalidades.
- Resolución excepcional (bajas oscilaciones).
- Muy bajo consumo de corriente (3.6 uA).
- Pequeño Tamaño (2.5 x 2.5 x 0.93 mm3).
- Tiempo de respuesta rápido.

**Desventajas**:

- Desfasaje de la temperatura debido a calentamiento interno del dispositivo.

Veamos ahora, la misma comparación, pero para la **Humedad Relativa.** En este caso**,** no hay tanta oferta como para temperatura. Los más utilizados de los de temperatura incluyen también un sensor de humedad. Buscamos que en el ambiente se pueda medir en todo el rango posible **(0%-100%)**, y según las normas mencionadas anteriormente sobre seguridad y

salud en el trabajo, la humedad ideal en un ambiente de trabajo, los valores deberían oscilar entre **50% y 70%.**

En la tabla se presentan los sensores de humedad para ver las especificaciones de cada uno y compararlos:

TABLA III: Análisis de Sensores de Humedad

| Humedad | Rango de Mediciones | Precision | Tiempo de Respuesta | Resolucion | Costo | Otros |
|---|---|---|---|---|---|---|
| Taller Mecanico | 0%-100% (ideal:50%-%70) | ±3% | Minimo posible | 0.1% | Minimos posible | "-" |
| DHT11 | 20%-80% | ±5% | 6 segundos | 1% | USD 2,50 | Digital. Es ademas Sensor de Temperatura |
| DHT22 | 0%-100% | ±2-5% | 2 segundos | 0.1% | USD 7,50 | Digital. Es ademas Sensor de Temperatura |
| BME280 | 0%-100% | ±3% | 1 segundo | 0.008% | USD 9,82 | Digital. Es ademas Sensor de Temperatura Y Presion. Muy rapida respuesta |

De nuevo, vemos una situación parecida a la anterior. Si bien el DHT22 cumple las características requeridas en el taller, el BME280 también las cumple y posee las ventajas mencionadas anteriormente. Además de poseer el rango completo de medición de humedad relativa y cumplir con el requerimiento de precisión, la resolución es significativamente menor, postulándose como mejor opción también como sensor de humedad. También, posee un tiempo de respuesta de **1 segundo**, siendo la mitad que el tiempo de respuesta del DHT22, y 6 veces menor que la del DHT11.

Para la **Presión Barométrica**, se decidió no analizar otros tipos de sensores, ya que las características necesitadas no dependen del lugar en donde se realicen las mediciones: es decir, estas mediciones son a fines informativos y no para poder cambiar los valores a valores ideales. Por ejemplo, en el caso de la temperatura, si detectamos que hace 5°C, subiremos la calefacción para que la temperatura se encuentre dentro de los parámetros ideales. Con la Presión, esto tiene grandes dificultades para su realización, necesitando equipos más sofisticados.

El sensor BME280 muestra la información de Presión Atmosférica y funciona como Altímetro, ya que existe una relación directa entre estas variables. Trabaja con un rango de mediciones de **300 hPa a 1100 hPa**, con una precisión de **+-1 hPA** y con una resolución de **0.18 Pa.**

Por lo tanto, el sensor elegido por las características anteriormente mencionadas es el **BME280**.
Toda la información fue extraída y esta detallada en el **Datasheet** de este sensor (ver Anexo).

### 3.1.2.2. Sensores de Gases

En este caso, nos centraremos en que sensores utilizaremos dependiendo el tipo de gases que queramos medir. Universalmente, los más utilizados en proyectos de Arduino son los sensores **MQ-X**.

Son sensores electroquímicos y varían su resistencia cuando se exponen a ciertos gases. Posee un calentador encargado de aumentar la temperatura del interior, pudiendo así reaccionar con estos gases provocando un cambio en el valor de la resistencia. La respuesta en estos sensores necesita tiempos prolongados para estabilizarse tras un cambio en la concentración. Son sensores que sirven para proyectos intermedios, pero cabe resaltar que no deberían ser usados para situaciones de alta complejidad.

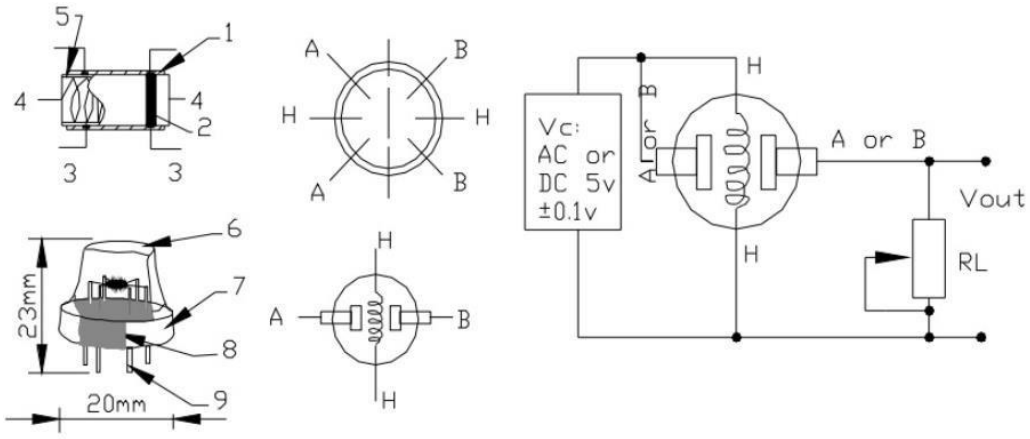



Figura 3.1.2.2.1: Esquema de Funcionamiento para cualquier sensor MQ

Generalmente, estos sensores pueden medir a más de un gas, pero lo que cambia, principalmente, es la sensibilidad a un gas más que a otro.


Figura 3.1.2.2.2: Sensores MQ-2, MQ-1 y MQ-3

Se suelen vender en módulos, permitiendo tener una salida digital utilizando un comparador y un potenciómetro, que determina la presencia o no de un gas.

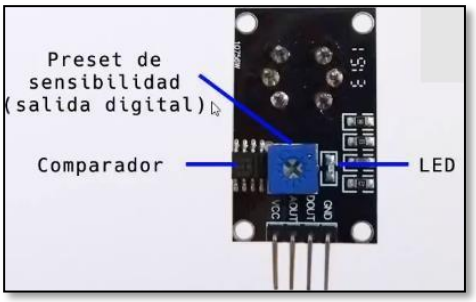



Figura 3.1.2.2.3: Configuración Digital de MQ-X

Lo importante en esta sección será definir que gases son los importantes a medir y, en base a ello, definir que sensores se usaran.

Para eso tenemos que definir qué tipos de gases son los que se emiten en la industria y pueden ser útiles para el proyecto. Vamos a tomar como supuesto distintos tipos de talleres en la i**ndustria automotriz** y las zonas de trabajo en ellas.

- **Mecánica**: Los motores utilizan Dióxido de Carbono, oxígeno y agua (gases no peligrosos) pero al ser procesados generan CO.
- **Refrigeración**: En equipos de aire acondicionado, se ha pasado de utilizar gases tóxicos como el R12 a otros menos contaminantes, como el R134.

- **Carrocería**: En la zona de soldaduras se utilizan hoy en día soldaduras con arco eléctrico (TIG, MIG, Plasma) y no oxiacetilénica, solo haciendo peligroso la inhalación de humo, pero no de carburos con oxigeno
- **Pintura**: La pintura utilizada ha bajado su emisión de gases considerablemente. Anteriormente los productos eran fabricados en base disolvente y plomo, haciendo las reacciones químicas perjudiciales.

Por supuesto, en todos estos sectores se poseen extractores de humo que evitan cualquier tipo de contaminación, pero esto podrá detectar fallas en el sistema o al menos servirá como control.

Podemos distinguir al menos 3 gases de los que debemos tener precaución: **Monóxido de Carbono, Humo y Gases inflamables.**


Figura 3.1.2.2.3: Sensores MQ-2 y MQ-7 y sus gases de mayor sensibilidad.

Hay varias combinaciones de sensores que se pueden realizar para poder cubrir esos 3 gases. En este caso, se eligió la combinación **MQ-2 y MQ-7**, que poseen su mayor sensibilidad a los gases mencionados anteriormente.

Los costos de estos sensores rondan los USD 4, siendo bajo dentro de la gama de sensores.

En caso de que se necesiten requisitos especiales por un tipo de industria diferente a la elegida, se podrá elegir cualquier otro sensor con mayor sensibilidad a otro tipo de gas: los costos son los mismos para todos los sensores MQ y la configuración es prácticamente la misma.

### 3.1.2.3. Sensor de Luminosidad

El sensor de luminosidad son dispositivos electrónicos utilizados para detectar la luz del ambiente, principalmente, la cantidad de luz brillo y oscuridad en el ambiente. Esa señal de luz, se traduce a señales eléctricas que permiten determinan esa cantidad de brillo detectado.

En el caso de los **foto-transistores**, convierten la luz en corriente o en voltaje. Los fotones entran en la base del transistor, generando los llamados "huecos" (espacios con carga positiva dejado por el movimiento de electrones) y una corriente que va colector a emisor. En este caso, los fotones funcionarían como la corriente de base de cualquier transistor general (ver Figura 3.1.2.3.1).

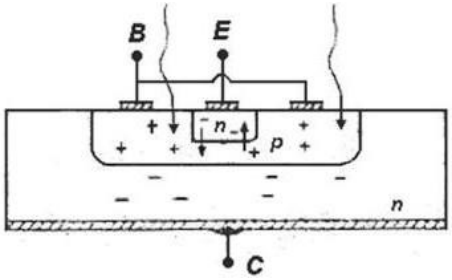



Figura 3.1.2.3.1: Esquema de un foto-transistor NPN.

En el caso de los **foto-resistores**, también llamados **LDR**, la resistencia decrece a mayor intensidad lumínica. Las radiaciones luminosas generan la energía que libera electrones, haciendo más conductor al material y así, disminuye su resistencia. Cuando es construida con sulfuro de cadmio, es sensible a todas las radiaciones luminosas visibles y con sulfuro de plomo, a las radiaciones infrarrojas.

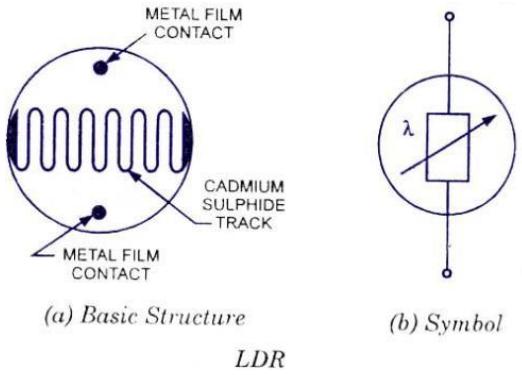



Figura 3.1.2.3.2: Estructura y simbología de un foto-resistor.

Aquí podemos ver algunos de los principales sensores utilizados con Arduino y los medibles que utilizamos en este caso para decidir qué modelo usar:

TABLA IV: Análisis de Sensores de Luminosidad

| Luminosidad | Temperatura Ambiente Requerida | Tipo | Tiempo de Respuesta | Resistencia | Rango Espectal | Costo |
|---|---|---|---|---|---|---|
| Taller Mecanico | 0ºC - 50ºC (ideal 14ºC-25ºC) | N/A | Minimo posible | Maximo rango posible | Maximo Posible | Minimo posible |
| PGM5506 | -30ºC a 70ºC | Foto Resistor | 30 ms | 2-6 Kohm | 540 nm | USD 0,27 |
| LM393 | 0ºC a 70ºC | Foto Resistor | 20 ms | 8-20 Kohm | 540 nm | USD 2,68 |
| TEMT6000 | -40ºC a 85ºC | Foto Resistor | 15 ms | 3.8-17.1 kOhm | 570 nm | USD 4,95 |

En esta categorización de los sensores de luminosidad más comunes, vemos que hay 2 foto-resistores y un foto-transistor. En el caso del fotorresistor LM393, este es un módulo que tiene dentro al LDR, con varios dispositivos extra, como un led de salida digital, los 4 pines ensamblados, un amplificador comparador, un led que indica alimentación y, lo más importante, un potenciómetro de ajuste de sensibilidad.

En cuanto a temperatura ambiente en la que funciona, vemos que todos cumplen perfectamente la condición, pero vemos que el LM393 no funciona a temperaturas bajo cero, punto que no es determinante para el uso de este proyecto, pero es un factor negativo respecto a sus competencias.

El mejor tiempo de respuesta es el del TEMT6000, que también gana en cuanto a temperatura de funcionamiento. Luego, el rango de resistencia determina que tan acertada van a ser las mediciones, ya que tiene mayor cantidad de valor sobre los cuales medir esta luminosidad, teniendo un rango 3.8-17 kOhm; también podemos ver que posee un mayor rango espectral, permitiendo entrada de luz de haces con longitud de onda mayores.

Definitivamente es el mejor en cuanto a sus características, pero su precio es significativamente más elevado que el del PGM5506, siendo casi 20 veces mayor. A pesar de esto, el LDR tiene un funcionamiento muy básico y si se le quisiera sumar los componentes que posee el módulo LM393, se haría más costoso (como vemos en el caso del módulo).

En conclusión, creemos que usar el foto-transistor **TEMT6000** traerá mejores resultados, simplicidad de utilización y mediciones más precisas en cuanto a datos de luminosidad, y por eso creemos que es el indicado para comprar.

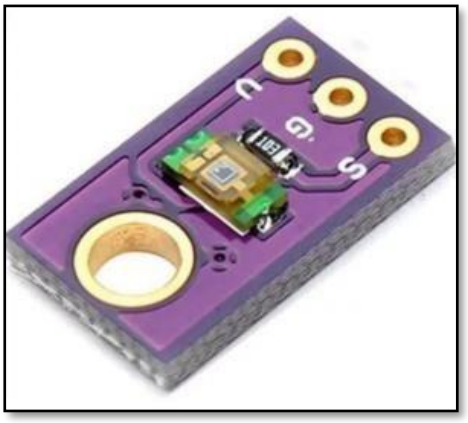


Figura 3.1.2.3.3: Modulo Foto-Transistor TEMT6000

### 3.1.3. Motores

Para la elección de los motores que iban a equipar al automóvil, investigamos sobre la oferta que existía para este tipo de proyecto. Al principio, consideramos el motor más utilizado en los proyectos de robótica con Arduino, el Dual Axis Gear Reducer 3V-6V. El mismo posee las siguientes características:

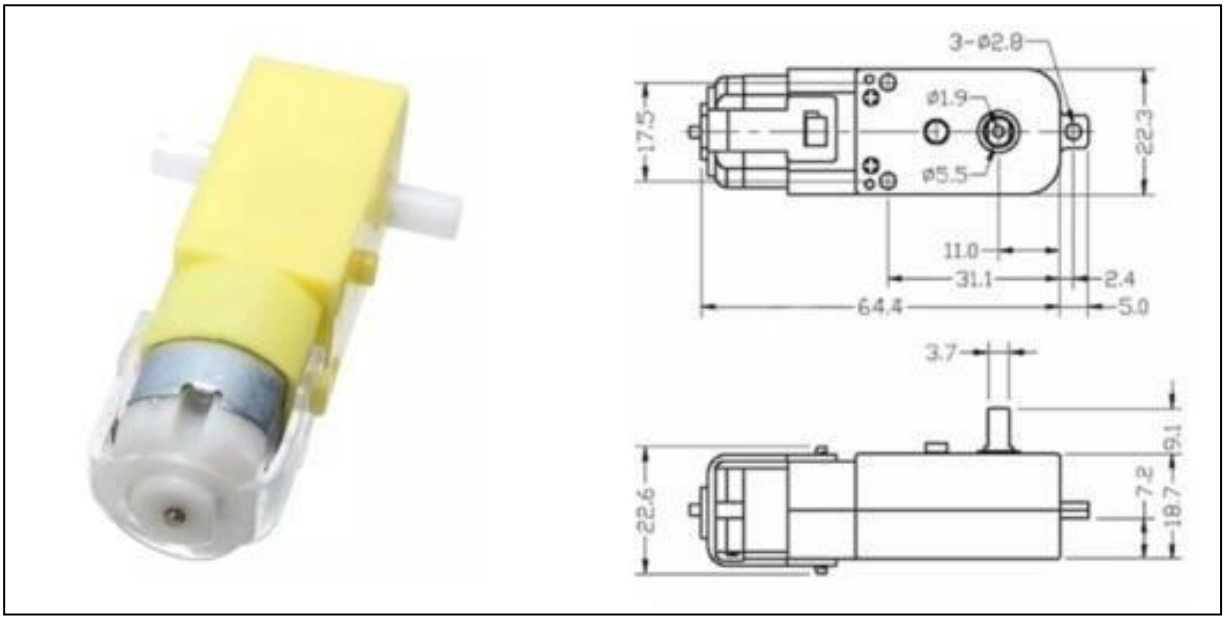



Figura 3.1.3.1: Motor Dual Axis

TABLA V: Ficha Técnica Dual Axis

| | Dual Axis Gear Reducer 3V-6V |
|---|---|
| Voltaje | 3V - 6V |
| Velocidad sin carga | 230 RPM |
| Velocidad con carga | 175 RPM |
| Torque de salida | 1.1 kg.cm |
| Corriente máxima | 130 - 150 mA |
| Peso | 50g |
| Ruido | <65dB |
| Dimensiones (LxD) | 70mmx22mm |
| Precio | U$S 1 |

Resulto ser un motor con buenas especificaciones a un muy bajo precio. Sin embargo, luego de gran cantidad de pruebas confirmamos que estos motores se encontraban muy limitados a

la hora de sortear pequeños obstáculos o pequeñas imperfecciones del suelo. Están diseñados para circular en superficies lisas y sin pendientes. Además, al estar fabricado en plástico, los posibles golpes que podría sufrir en el uso podrían dañar los motores permanentemente.
Fue así, que escogimos un motor de mayor potencia y de una aleación de aluminio más resistente a los golpes. El **Motor de Engranaje** de 25mm y 9v.

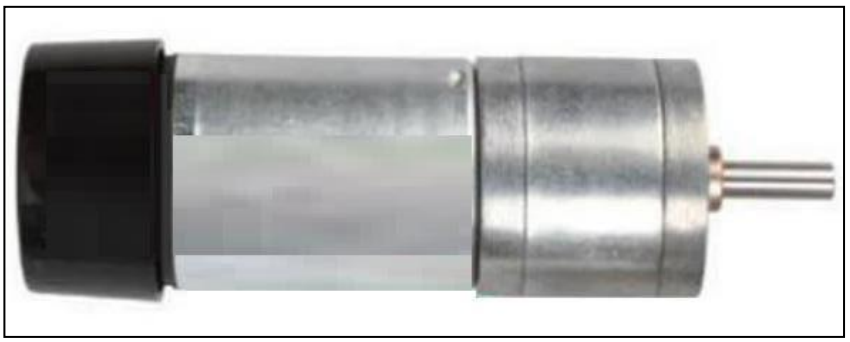


Figura 3.1.3.2: Motor de Engranaje

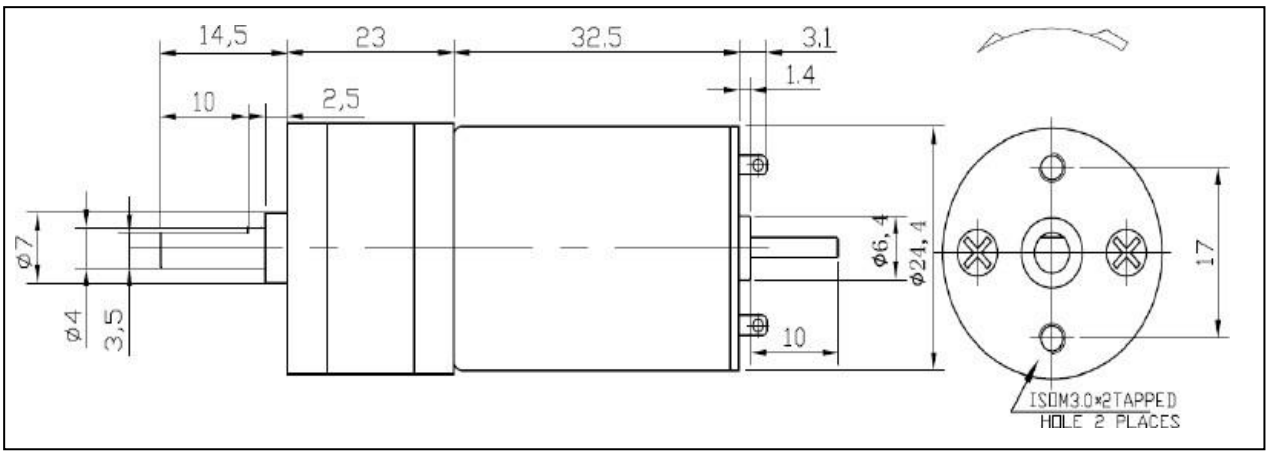



Figura 3.1.3.3: CAD Motor de Engranaje

TABLA VI: Ficha Técnica Motor de Engranaje

| | Motor de Engranaje 3-9V con Sensor Hall |
|---|---|
| Voltaje | 3V - 6V |
| Velocidad sin carga | 150 RPM |
| Velocidad con carga | 110 RPM |
| Torque de salida | 9.5 kg.cm |
| Corriente máxima | 1200mA |
| Peso | 70g |
| Ruido | <56dB |
| Dimensiones (LxD) | 70mmx24.4mm |
| Precio | U$S 2 |

Estos nuevos motores, por una diferencia de costo muy baja, permitieron que el robot se moviera con más libertad superando los pequeños obstáculos que se le presentaban en el camino. Esto fue gracias a su diferencia notable de entrega de torque. Además, estos motores, cuentan con sensor Hall que, aunque en este proyecto no se utilizó, sirven para saber la orientación y posición del motor basado en la detección o medición de campos magnéticos o corrientes (Efecto Hall).

### 3.1.4. Chasis

Para la selección del chasis evaluamos diferentes alternativas. Debíamos tener en cuenta que el robot tendría que ser estructuralmente resistente y capaz de sobrepasar pequeños obstáculos. En primera instancia, analizamos la opción de un chasis equipado con ruedas. El primer modelo que evaluamos fue el **4WD Smart Robot**. El mismo está pensado para 4 motores ensamblado en una estructura de acrílico.

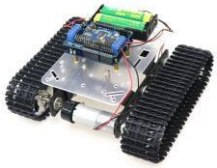
Figura 3.1.4.1: 4WD Smart Robot

Este chasis fue descartado rápidamente ya que, estructuralmente no era robusto y, además, presentaba problemas de tracción.

Fue de esta manera, que evaluamos la posibilidad de cambiar la tracción a la de **orugas**. Buscamos una manera de que el robot se moviera de forma eficiente y precisa en todo tipo de entornos. Como el aspecto más crítico era el sistema de locomoción, veíamos a las orugas como una alternativa. Este sistema usa pistas de deslizamiento, lo que implica una mayor área de contacto con el terreno, y por tanto supone una mejor maniobrabilidad, mejor tracción que las ruedas y una movilidad superior. Además, solo requiere dos motores para tracción y a diferencia de las ruedas no requiere sistema de suspensión. Las orugas generan una baja presión en el suelo, y reparten el paso de una mejor manera. Otra ventaja se encuentra en que el centro de gravedad se mantiene más bajo, lo que lleva al vehículo a tener una mejor estabilidad y movilidad. El radio de giro también es un punto a favor, los giros pueden ser excesivamente cerrados o hasta con radio de giro igual cero.

Con este criterio evaluamos otras opciones fabricadas con diferentes materiales y de tracción a orugas.

TABLA VII: Opciones de Chasis de Orugas

| | Opciones de Chasis de ORUGAS | | |
|---|---|---|---|
| Nombre | Tanque Chasis Diy Kit | Chasis T100 | TS100 Metal Chasis |
| Foto |  |  |  |
| Tamaño (ancho * largo * alto) | 9,7 cm * 18,5 cm * 5 cm | 18,5 cm * 20,0 cm *6,0 cm | 27,5 cm * 19 cm * 9,5 cm |
| Material | Plástico | Aleación Aluminio | Aleación Aluminio |
| Peso | 0,4 kg | 0,65 kg | 1,10 kg |
| Transmisión | A orugas | A orugas | A orugas |
| Cantidad de motores | 2 | 2 | 2 |
| Costo (U$S) | U$S 11 | U$S 35 | U$S 70 |

De las alternativas analizadas, decidimos descartar el fabricado en plástico por su baja resistencia a impactos. De las otras dos alternativas, nos encontramos con dos chasis muy similares, pero de diferente tamaño, con mismas piezas y materiales. Consideramos innecesario pagar el doble de precio para aumentar solo en tamaño. Optamos por la versión **T100**; tiene un chasis robusto de aleación de aluminio con un tamaño que nos permite ubicar el micro controlador, a alimentación y los sensores de manera cómoda y segura.

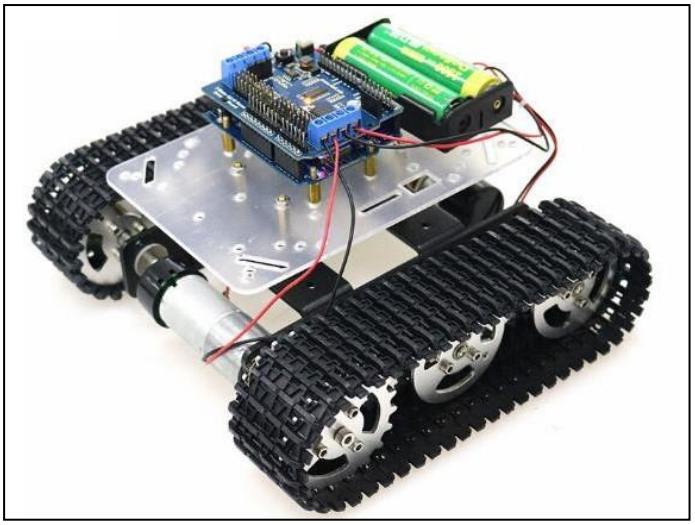


Figura3.1.4.2: Chasis T100

### 3.1.5. Protocolo de Comunicación

El **protocolo de comunicaciones** se refiere a la forma en la que nuestro dispositivo se comunicara, de manera inalámbrica, con el usuario y como se transmitirán y recibirán los datos de los sensores. Hoy en día hay amplia variedad de opciones para los tipos de conexión y módulos que se pueden conseguir fácilmente para ampliar la conectividad. Se debe elegir el tipo de comunicación que más conveniente para el proyecto, y el tipo de módulo o dispositivo que utilizaremos para implementarlo.

Algunos criterios son necesarios para poder comparar entre los distintos tipos de comunicación. Consideramos que las siguientes son las más importantes a la hora de realizar nuestro proyecto:

- **Distancia de Uso**: Es la distancia máxima que puede haber entre el usuario al transmitir y recibir los datos, y el dispositivo en cuestión.
- **Consumo**
- **Conectividad**: Forma de conexión entre y facilidad de realizar esta conexión.
- **Periféricos de Comunicación**: Con que periféricos puedo realizar el envío y recepción de la información.
- **Facilidad de Implementación y Utilización**
- **Costo**: Costo del módulo que adapta esta conectividad a Arduino.
- **Adicionales**: Detalles adicionales no contemplados anteriormente.

Antes de empezar la comparación, mencionaremos los **dos** tipos de comunicación elegidos para comparar que se estarán analizando en esta sección:

**Bluetooth:**

Se le llama Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, bajo alcance de emisión y que está basado en transceptores de bajo costo. Todo tipo de comunicación se realiza por radiofrecuencia. Está compuesto por 2 partes de hardware importantes: un dispositivo de radio (encargado de modular y transmitir la señal, y un controlador digital, compuesto por una CPU, DSP e interfaces. Los dispositivos

Bluetooth, se suelen clasificar por su ancho de banda, que fue aumentando desde las Versiones 1.2 (1Mbit/s) hasta las más novedosa Versión 5 (50 Mbit/s).

Figura 3.1.5.1: Logotipo de los dispositivos Bluetooth

Bluetooth opera en la banda de frecuencias no regulada (bansa ISM) con una frecuencia de 2.4 Hz. Los dispositivos con Bluetooth pueden actuar como Masters (maestros) o Slaves (esclavos); los maestros pueden conectarse a otros esclavos (máximo 7), simultáneamente, mientras que los esclavos solo pueden conectarse a un maestro. Si el emparejamiento entre ambos nodos se realiza con éxito, los dispositivos tienden a guardar la identificación del otro para una conexión automática.

El módulo más utilizado en Arduino para darle conectividad Bluetooth es el **HC-05**. Este módulo posee Bluetooth V2 y es realmente económico, junto con el HC-06. También existen módulos con Bluetooth LE (lowenergy), que reducen el consumo, pero también la distancia de uso, así que no los vimos convenientes de analizar.


Figura 3.1.5.2: Modulo HC-05 FC-114 para conexión Bluetooth

**Wifi:**

El Wifi es una tecnología utilizada hoy en día en todo el mundo, que permite a diferentes dispositivos conectarse entre sí o a internet, uniéndose a un punto de acceso de red

inalámbrica. Es una marca de la Alianza Wi-Fi, y cumple con los famosos estándares 802.11 Opera de la misma manera que el bluetooth; en una banda de 2,4 GHz (también existe su versión más moderna que opera en la banda de los 5Ghz).

Figura 3.1.5.3: Logo de los dispositivos Wifi

Existen diversas formas de su conexión con el Arduino, pero la más utilizada es el chipset ESP8266. Es un microprocesador de bajo coste con Wifi integrado, que se puede utilizar para conectar Arduino con diferentes dispositivos Wifi. En un inicio existían otros módulos (Wifi Shield) de Arduino, pero costaban demasiado y el ESP8266 fue el chipset que permitió proyectos de bajo costo con conectividad incluida.

Muchos módulos poseen el ESP8266 integrado, creando adaptadores de su tecnología que pueden ser utilizados para Arduino, pero el más común y barato de estos es el ESP01. Tiene un procesador de 32bits a 80 Mhz, con 512kB o 1MB según los distintos modelos. Incluye Wi-Fi Direct (Peer-to-Peer) y una pila de TCP/IP completa.

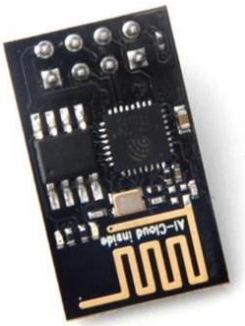


Figura 3.1.5.4: Modulo ESP01 Wifi

La comparación realizada fue la siguiente:

- **Conectividad**: El Bluetooth posee una conexión directa, que es lo necesario para este proyecto; el Wifi necesita un **punto de acceso a la red**, que puede o no estar disponible al utilizar el automóvil.

- **Consumo**: Los módulos de conectividad Wifi dan una gran carga para el procesamiento de Arduino. Otros dispositivos como Raspberry Pi o el mismo ESP8266 pueden servir para este propósito, pero no son los utilizados y elegidos en un principio para este proyecto. El ESP01 consume 200mA (hay que tener en cuenta que la salida 3.3V de Arduino no puede enviar más de 60 mA). El módulo HC-05 consume 50ma.
- **Facilidad de Implementación y Comunicación**: Requiere muchas más configuraciones (conexión a explorador Web, liberación de puertos, conocimiento de IP del módulo, etc.) para su conectividad que no son relevantes o necesarias para este proyecto.
- **Periféricos de Comunicación**: La mayoría de los dispositivos poseen los dos tipos de conexiones; teléfonos móviles, tablets, notebooks, etc.
- **Distancia de Uso**: Los proyectos que utilizan conexión Wifi suelen ser aquellos para los cuales se requiere una conexión a larga distancia (Ej.: un sistema de alarmas para un hogar). Para los proyectos de cortas distancias, la conexión Bluetooth es suficiente para los propósitos necesitados (15m y los modelos clase 1, 100 m, a mayor consumo).
- **Costo**: Los costos son muy similares, teniendo un costo levemente mayor el HC-05, pero no siendo significativo en el análisis.

Debido a estas razones, para este proyecto, elegimos utilizar la comunicación **Bluetooth,** con el módulo **HC-05,** en vez de la conectividad Wifi.

Más allá de haberlo elegido, existen diversos protocolos de comunicación que no han sido analizados y poseen un más largo alcance, pero por diversas razones (entre ellas complejidad y costo), no fueron elegidos. En este diseño, nos concentraremos en un modelo de **corto alcance**, siendo escalable a rangos mayores por medio de otros protocolos de comunicación. Si se requiriese de un alcance mayor, Bluetooth no sería el protocolo ideal.

### 3.1.6. Otros Accesorios y Herramientas

Mas allá de todos los componentes como sensores, actuadores y partes de la estructura, existen diversos elementos que se utilizan para el conexionado, para la conversión de niveles lógicos, entre otras funciones. Para realizar esta lista, tomamos en cuenta los siguientes puntos:

- Particularmente, para el testeo de varias funciones del prototipo y para la investigación, utilizamos otros accesorios que **NO** incluiremos en la lista (protoboards, LEDs, etc) debido a que no forman parte del producto final. Solo fueron utilizados confines de testeo.
- Incluimos el **Herramental**, para dar noción acerca de que herramientas y elementos se deben tener para poder hacer el proyecto. Este costo no fue tomado en cuenta.
- Algunos de los elementos de la lista, fueron comprados en grandes cantidades ya que son consumibles de bajo costo. Para incluirlos en este listado, colocamos el costo de la compra realizada, más allá de que se hayan usado todos o no.
- En la lista de Accesorios se pondrán los rótulos
    - **Nombre**
    - **Descripción**: Se dará a conocer el uso que se le dio en proyecto al accesorio o herramienta.
    - **Imagen**: Imagen para dar una noción más detallada del accesorio utilizado
    - **Cantidad Comprada (**Accesorios**)**: Cantidad que enviamos a comprar, pero que no necesariamente fue utilizada.
    - **Precio:** Precio de la cantidad comprada.
- Para la lista de Herramental, se tomarán solamente los primeros 3 rótulos descriptos anteriormente.

TABLA V: Listado de Otros Accesorios

| Otros Accesorios | | | | |
|---|---|---|---|---|
| Nombre | Descripcion de uso | Imagen | Cantidad Comprada | Precio (U$D) |
| Cables Dupont de Alambre H-H | Cables para la realizacion de conexionado de pines de tipo Hembra-Hembra |  | 40 unidades | USD 2,05 |
| Cables Dupont de Alambre M-M | Cables para la realizacion de conexionado de pines de tipo Macho - Macho |  | 40 unidades | USD 2,05 |
| Cables Dupont de Alambre H-M | Cables para la realizacion de conexionado de pines de tipo Hembra-Macho |  | 40 unidades | USD 2,05 |
| Level Shifter 3.3V- 5V | Cambiador de nivel logico utilizado en el sensor BME280 ya que este funciona con 3.3V y no con 5V |  | 1 unidad | USD 1,25 |
| Arduino Mega Sensor Shield V2.0 | Escudo para el Arduino Mega que permite el conexionado de sensores de una manera mas comoda, sin necesitar otros elementos de conexion. Posee pines de alimentacion distribuidos a lo larga de la |  | 1 unidad | USD 5,36 |
| Estaño para Soldadura | Estaño utilizado para soldadora. |  | 1 metro | USD 3,00 |
| Pines para Soldar | Pines utilizados para que con la soldadora, se adhieran a algunos componentes que no vienen con pines salientes incorporados. |  | 40 pines (tira) | USD 0,60 |
| Placa Acrilico | Placa utilizada para sostener al Arduino y algunas de sus conexiones en la superficie superior del auto. | | | |
| Holder 4 Pilas AA | Holder de 4 Baterias en Serie, utilizadas para la alimentacion del dispositivo. |  | 1 unidad | USD 2,32 |
| | | | **Total** | **USD 18,68** |

TABLA VI: Listado de Herramental y Software

| Herramental y Software | | |
|---|---|---|
| Nombre | Descripcion de uso | Imagen |
| Computadora / Notebook | Lo mas utilizado para este proyecto. Permitira programar el codigo, hacer los diagramas, la interfaz de usuario, el informe, etc. |  |
| Conexion Wi-Fi | Es imprescindible para la busqueda de informacion, pero particularmente para el acceso a la plataforma de creacion de interfaz de usuario (MIT App Inventor 2) |  |
| Android Smartphone | Se utilizara para testear la interfaz de usuario y para la utilizacion del dispositivo. |  |
| Software IDE "Arduino" | Software utilizado para programar y cargar el codigo al microcontrolador en lenguaje Arduino (basado en C++). |  |
| Software "Fritzing" | Software utilizado para diagramar las conexiones entre los componentes. |  |
| Soldador de Estaño | Utilizado para soldar los pines a algunos de los componentes que no vienen con ellos integrados. |  |
| Destornillador con Punta Removible | Utilizado para el ajuste de diversos tornillos. |  |
| Llaves Allen | Utilizadas para el ajuste de tuercas. |  |
| Cable USB Tipo B | Para conectar el microcontrolador a la computadora. |  |

### 3.1.7. Costos

Para tener una noción del costo total del proyecto, tomamos en cuenta todos los componentes seleccionados anteriormente, y sumamos el precio de cada uno de ellos. Luego, mediante algunos gráficos estadísticos, mostraremos en donde se centran los costos de esto.

#### 3.1.7.1. Criterios

Planteamos ciertos criterios a la hora de contabilizar los costos:

1. Toda contabilización se realizó en **U$D (Dólar Americano)**, para dar un valor unificado internacionalmente.
2. Se tomó un **Tipo de Cambio Fijo** entre **ARS (Peso Argentino)** y el dólar, en un valor de **ARS 56,02 = U$D 1**. Esto fue tomado en el momento de realizada la primera compra para el proyecto, para evitar el desbalance provocado por las contantes fluctuaciones del tipo de cambio en nuestro país.
3. Los precios de los componentes fueron obtenidos de la siguiente forma:
    **a.** Se buscaron proveedores en páginas web (MercadoLibre, OLX).
    **b.** Se saco un promedio entre los valores encontrados para cada componente, en ARS.
    **c.** Ese valor, con el tipo de cambio fijado anteriormente, fueron pasados a dólares y ese es el valor utilizado para contabilizar.
4. Para los únicos componentes que **NO** fue conseguido el precio de esta forma, fue para los Motores y el Chasis, ya que estos productos eran muchísimos más baratos en China (por medio de AliExpress) que, en Argentina, así que la decisión que tomamos fue importarlos. El precio estaba directamente en dólares, así que no se realizó conversión.
5. Basados en nuestra premisa de que el proyecto debería ser **económico**, pusimos un precio máximo basado en otros dispositivos similares (algunos sin sensores) y

logrando una reducción importante el precio. El precio que fijamos para el proyecto es de **USD 130.**

### 3.1.7.2. Tabla y Gráficos

TABLA VII: Tabla de Costos Totales

| Categoria | Sub-Categoria | Modelo | Cantidad | Costo p /Unidad (U$D) | Costo Total (U$D) |
|---|---|---|---|---|---|
| Microcontrolador | Microcontrolador | Arduino Mega 2560 + Cable USB | 1 | USD 21,00 | USD 21,00 |
| Sensores | Sensor de Temperatura, Humedad y Presion | Modulo Sensor BME280 | 1 | USD 9,75 | USD 9,75 |
| Sensores | Sensor de Gas | Modulo Sensor MQ-02 | 1 | USD 4,00 | USD 4,00 |
| Sensores | Sensor de Gas | Modulo Sensor MQ-07 | 1 | USD 4,00 | USD 4,00 |
| Sensores | Sensor De Luminosidad | Modulo Sensor TEMT6000 | 1 | USD 4,95 | USD 4,95 |
| Protocolo de Comunicacion | Modulo de Conectividad | Modulo HC-05 Bluetooth | 1 | USD 3,57 | USD 3,57 |
| Motores | Motores | Motor de Engranaje 3-9V con Sensor Hall | 2 | USD 2,00 | USD 4,00 |
| Chasis | Chasis y Transmision | Chapas,Engranajes, Oruga y Tornilleria | 1 | USD 31,00 | USD 31,00 |
| Otros Accesorios | | | - | USD 18,68 | USD 18,68 |
| | | | | **Total** | **USD 100,95** |

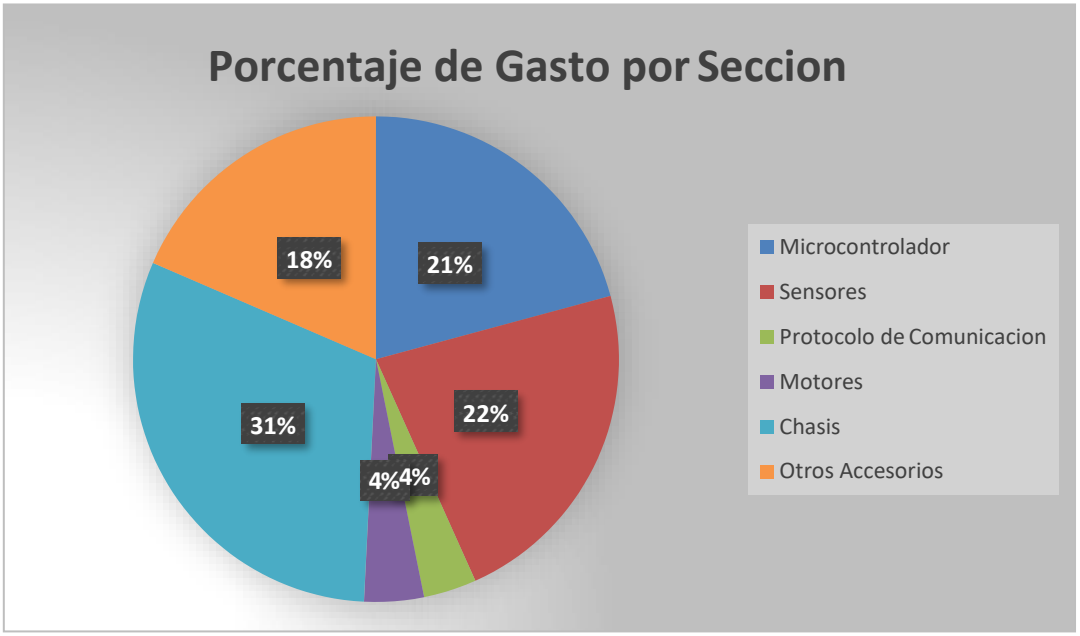



Figura 3.1.7.1: Grafico de Torta por Sección de Proyecto

En este grafico podemos ver el porcentaje del gasto que representa cada sección del proyecto. Vemos que está bastante distribuido entre las distintas partes del proyecto, sin tomar en cuenta el protocolo de comunicación y los motores, que son de bajo costo.

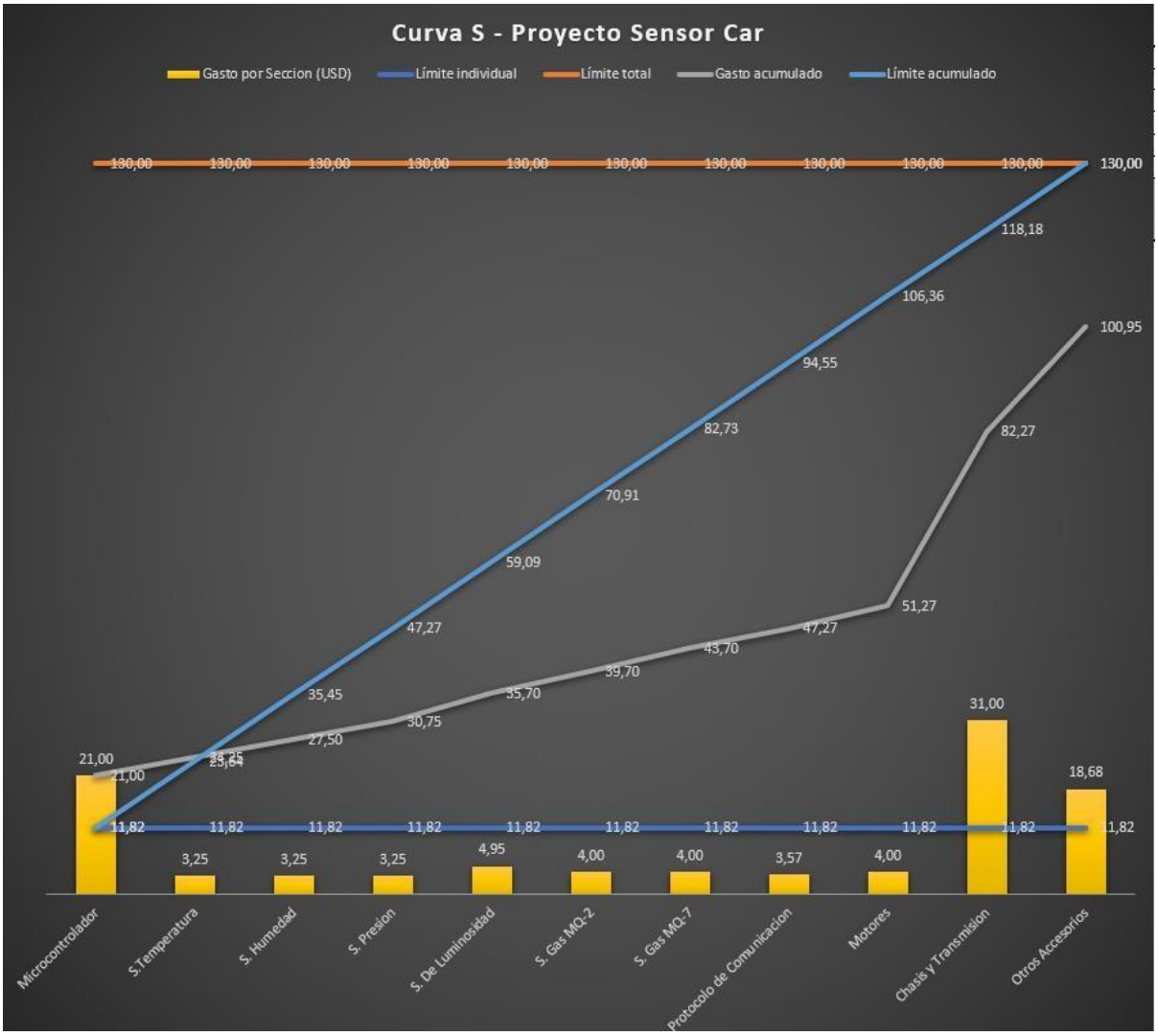



Figura 3.1.7.2: Curva S por Sección de Proyecto

La Curva-S nos permite ver cómo, a medida que vamos avanzando en proyecto, vamos utilizando nuestro presupuesto. Esta curva, supone una distribución por igual en cada etapa del proyecto del presupuesto inicial (definido anteriormente como USD 130) y a medida que vamos avanzando en cada etapa, deberemos ir compensando el déficit en un área, con el superávit de otra. Es utilizada para un análisis a tiempo real, y para ir haciendo un seguimiento.

Al final de la curva, podemos ver que el valor acumulado no supera el presupuesto inicial, teniendo así cumplido nuestro objetivo de gastar un monto menor a USD 130.

## 3.2. Implementación y Metodología

Luego de la revisión realizada acerca de los componentes que utilizaremos en nuestro proyecto, explicaremos como fue que estos fueron implementados en el proyecto y como fueron pensados para ser utilizados.

### 3.2.1. Diseño Electrónico

El diseño de las conexiones fue realizado con el programa Fritzing, un programa especialmente utilizado para este tipo de diseños, ya que posee muchísimas piezas que pueden ser utilizadas para los diagramas. Incluso, los usuarios pueden subir sus propias piezas y se hace una red de partes muy útil para la creación de circuitos complejos o con elementos muy específicos.

Figura 3.2.1.1: Logotipo del programa utilizado Fritzing

En el proyecto estará adjunto el archivo del programa Fritzing en el ANEXO. Por ahora, veremos con una captura del circuito las diferentes partes y sus conexiones para el entendimiento general.

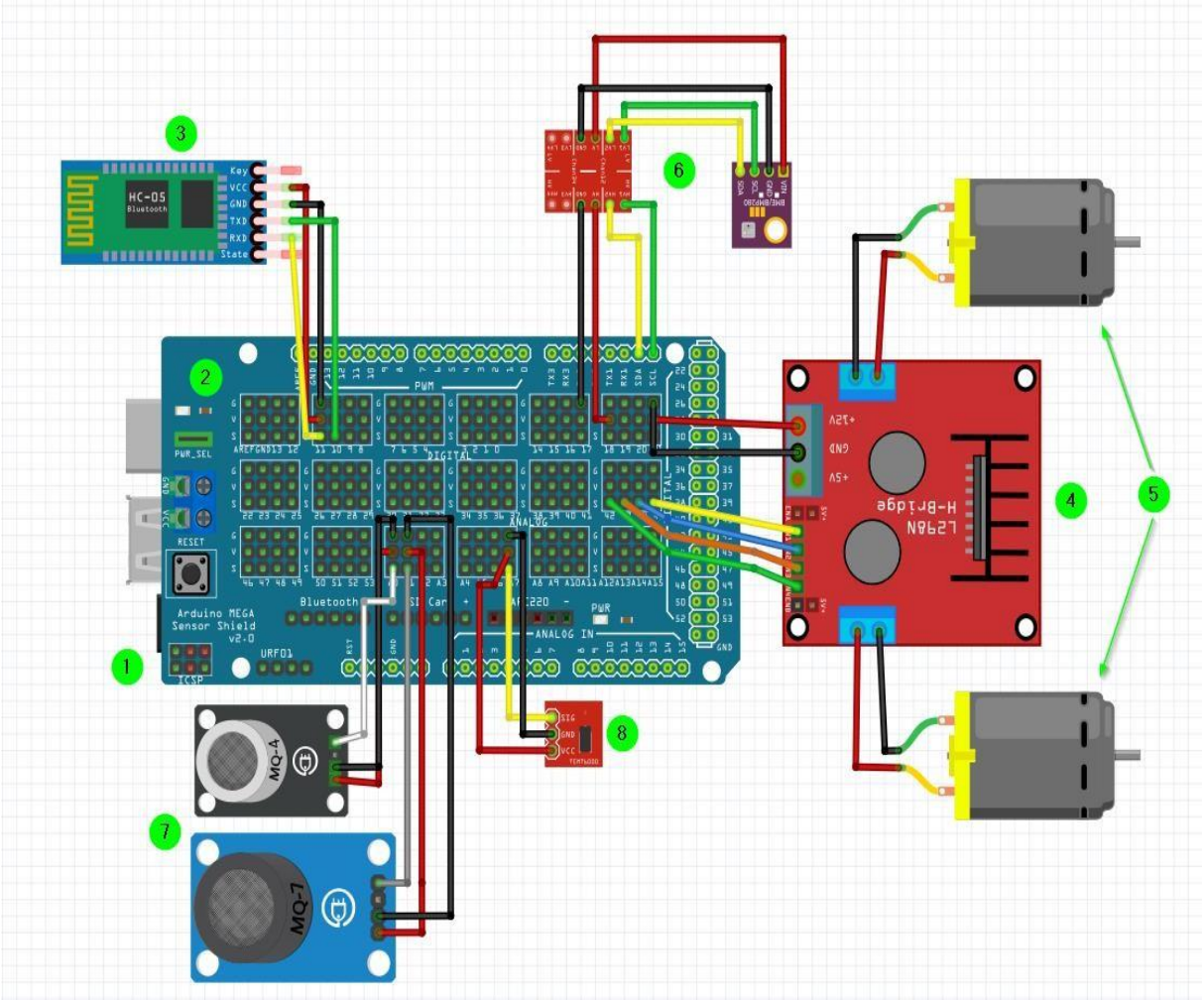



Figura 3.2.1.2: Captura del circuito realizado en Fritzing

En el esquema fue marcado con números las distintas partes a explicar:

1. **Arduino Mega 2650:** El microcontrolador elegido es la parte que controla todos los sensores y actuadores conectados. Los pines envían y reciben señales de los diferentes periféricos. Este está conectado a un Sensor Shield que será mostrado en el próximo punto.
2. **Arduino Sensor Shield V2:** Este shield fue elegido por su capacidad de simplificar circuitos, como vimos anteriormente. En el diagrama, este reprograma la ubicación de los pines, además de distribuir la alimentación de los pines para facilitar su conexión. En este caso, el sensor también cambia los pines de ser hembra, a macho (alambre salido al exterior). Por cada salida/entrada digital o analógica, salen 3 pines que representan la

alimentación positiva (V), la tierra (GND) y el pin de señal (S), como se puede apreciar en la Figura 7.2.1.3. Esto facilita ampliamente la conexión de cada sensor.


Figura 3.2.1.3: Pines del Arduino Mega Sensor Shield

3. **HC-05 Bluetooth Module**: Este módulo es el protocolo de comunicación elegido que posibilita la transferencia de datos por Bluetooth entre la Aplicación de celular creada, como explicamos en la Selección de Componentes. En su conexión se utilizan los 4 pines centrales, no siendo necesario utilizar los pines de *State* y *En*, de los bordes, que sirven para configuraciones más avanzadas.

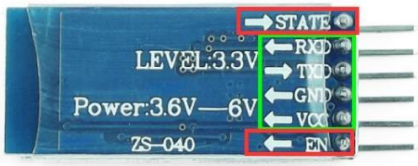



Figura 3.2.1.4: Conexiones para Modulo HC-05

Las salidas RXD (recibo de datos) y TXD (envió de datos), deben estar conectadas a **la inversa** con sus respectivas en Arduino. En este caso, decidimos no superponer la transferencia de datos de Arduino y el Serial, conectándolo a los pines RXD y TXD de Arduino; sino que, con una biblioteca en C++, programamos para que el envío de datos se realice a dos pines comunes digitales, programados para recibir datos. Mas adelante en el informe mostraremos como hace la Aplicación de Android, para conectarse a este dispositivo.

4. **DC Motor Driver L298N:** En este punto se encuentra el driver de los motores conectados a Arduino. Esto permite que desde Arduino se manejen las direcciones de cada uno de los motores. Se controla la lógica de este dispositivo en los pines designados como **IN1, IN2, IN3 e IN4.**Los primeros dos pines controlan al primer motor y los otros dos al segundo motor; dependiendo de las combinaciones de ellos, la dirección y sentido del auto cambiara. También, este controlador debe recibir la alimentación que será proporcionada a los motores, y esta será abierta cuando alguno de los dos pines IN de cada motor, esté

encendido. Si los dos mandan un valor 0, este motor no arrancara. Esto será visto en detalle en la sección de Alimentación.

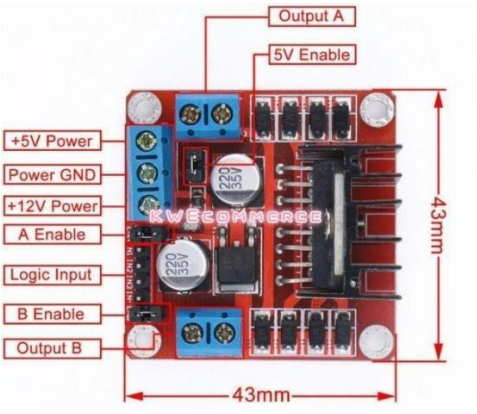



Figura 3.2.1.5: L298N Motor Drive en detalle

Los pines marcados en la Figura 7.2.1.5 como “Output A” y “Output B”, son los pines que van conectados a alimentar los motores. Este driver da la capacidad de ser alimentado por 5V (directo de Arduino), conectando el positivo a uno de los pines o hasta 12V, conectándolo a otro de los pines. Esto permite ser más flexible y con este mismo driver poder controlar cualquier tipo de motor.

5. **DC Motor 9V:** En el esquema electrónico, estos motores no están representados tal cual son, pero la conexión es la misma; la salida del driver va directo al positivo y negativo de las fichas del motor. Particularmente, los motores elegidos son de 9V; a mayor voltaje, mayor velocidad. En la sección de Alimentación, veremos cómo optamos alimentarlos. Un detalle importante de estos motores es que poseen un **Sensor de Efecto Hall.** Esto, en resumidas cuentas, mide el campo magnético dentro del rotor y se utiliza principalmente para medir la velocidad con la que está girando el motor.

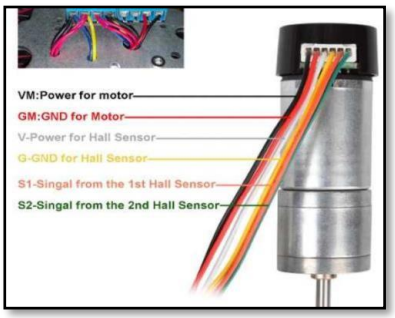



Figura 3.2.1.6: Pines para Motor con sensor de Efecto Hall

En este proyecto no vimos necesaria la utilización de este sensor, ya que complicaría las conexiones solo para proporcionarnos de un dato que no es necesario para este trabajo.

6. **Sensor BME280 + LevelShifter 3.3V-5V:** Aquí vemos el sensor anteriormente analizado; el BME280, utilizado para medir Temperatura, Humedad y Presión. Este sensor, se comunica con Arduino a través del bus conocido como **I2C.** Es un modelo creado en 1982 para la comunicación entre dispositivos electrónicos y sus periféricos. Solo requiere dos conexiones para su funcionamiento; la conexión de la señal del reloj (SCL) y uno para el envió de datos (SDA). Utilizando este bus, cada dispositivo tiene una dirección que es fijada por Hardware. En el caso de este sensor es la dirección en hexadecimal 0x76. Es una dirección única, que puede ser cambiada solo mediante cambios de hardware en el dispositivo a conectar.

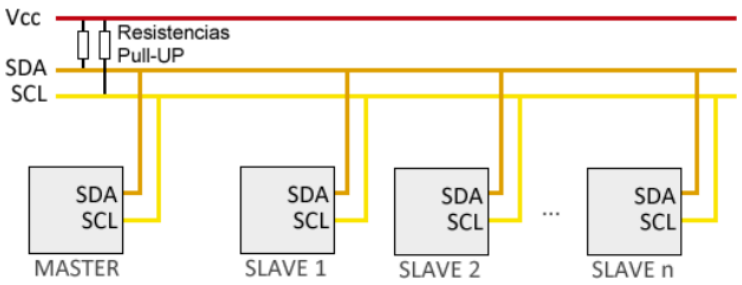



Figura 3.2.1.7: Conexión I2C entre Maestro y Esclavos

Como podemos ver en la Figura 7.2.1.7, el dispositivo posee una arquitectura de maestro-esclavo; es decir, que el maestro inicia la comunicación y los esclavos solo envían datos y reciben datos sin comunicarse entre sí. Gracias a la señal de reloj enviada por el maestro, todos los dispositivos conectados por este bus están sincronizados. Generalmente requiere resistencias de *Pull-up* de las líneas a Vcc, pero estas pueden ser reemplazadas mediante software (Librería Wire.h), que posibilita la activación de las resistencias internas del dispositivo.

Explicado cómo funciona el I2C, sabemos que entonces las conexiones del BME280, precisaran solamente la conexión de los pines SCL y SDA.


Figura 3.2.1.8: BME280

El problema con este sensor, es que utiliza 3.3V para su funcionamiento, tanto de los pines digitales como de alimentación, en vez de los 5V a los que estamos acostumbrados. Lamentablemente, el Arduino Mega Sensor Shield, tiene soldado el pin de 3.3v que sale del Arduino Mega, haciendo imposible una conexión más simple. Por eso conectamos un Controlador de Nivel (LevelShifter) que posibilita que todos los pines, tanto como la alimentación que este a 5V, tengan una salida a 3.3V, posibilitando así el funcionamiento del sensor.

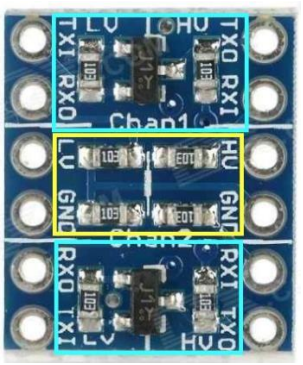



Figura 3.2.1.9: LevelShifter 3.3V- 5V con pines marcados

Los pines marcados en celeste en la Figura 7.2.1.9, son los que transfieren datos, y los marcados en amarillos, transfieren VCC y GND. La parte que dice LV (LowVoltage) es la que va conectada al sensor BME280 y la parte que dice HV (High Voltage) es la que va conectada al Shield de Arduino.

7. **Sensores MQ-2 y MQ-7:** En el esquema electrónico vemos conectados estos sensores (el MQ-2 fue reemplazado por un MQ-4 ya que no había una pieza del MQ-2). La conexión de estos sensores es bastante simple, pero hay que tener en cuenta que tipo de conexión se quiere tener. Esta gama de sensores posee 4 pines: uno de alimentación Vcc, tierra GND, uno de salida analógica y otro de salida digital. Para la salida digital, en la parte trasera,

posee un potenciómetro incorporado que permite mediante su movimiento, la configuración de la cantidad de gas de referencia (marca en verde en la Figura 7.2.1.10). Esto permite que el sensor envié una señal cuando se pasa de ese umbral.


Figura 3.2.1.10: Parte Trasera de Sensor de Gas MQX

De todas formas, para este sensor elegimos usar la **salida analógica**, y conectarlo a un pin de estas características en el Arduino Mega 2560, para poder generar una medición de la cantidad de gas. Si bien la conexión es simple, la programación para poder tener la cantidad de Gas en el ambiente, tiene sus complicaciones y será visto en la sección de Programación.

8. **Sensor de Luminosidad TEMT6000:** El sensor de luminosidad elegido anteriormente, no posee grandes complejidades a la hora de su conexión. Simplemente, se conecta la señal de salida a un pin analógico de Arduino y se leen los datos proporcionados por el sensor. El programa se encarga de interpretar estados datos y convertirlos a un valor porcentual.

### 3.2.2. Alimentación

Existen diversas formas de alimentar un proyecto Arduino. Para entender cuál es la mejor de ellas, es importante entender el hecho de que siempre estamos trabajando a **5V**, en la placa, y podemos trabajar en un voltaje mayor en los motores, pero no será necesario. Esto se debe a que los motores solo varían su velocidad con el voltaje, y no necesitamos velocidad para este proyecto. Por eso, nuestro enfoque para la alimentación estará basado en un esquema a **5V y daremos prioridad a la placa.**

**Mecanismos de Alimentación:** Existen tres mecanismos de alimentación en una placa Arduino:

- **Puerto USB**: La forma más sencilla de alimentar. Se admiten únicamente **5V**. Por este puerto, se puede alimentar el proyecto conectándolo a una computadora, a un cargador de pared o a un cargador portátil de emergencia. Posee un fusible tipo PPTC, que limita la corriente de salida a 500mA. Por computadora o con cargador de pared no se puede hacer la carga ya que el proyecto es de categoría portable, y la carga limite es muy baja para este tipo de proyectos, limitando el uso del puerto únicamente para cargar programas a la placa.
- **Jack de Alimentación**: Tipo de alimentación estándar en muchos dispositivos, con polo positivo en el centro. Acepta entre 7V -12V y la placa lo transforma en 5V. Para trabajar en el punto más óptimo y que no se sobrecaliente el regulador, se debe optar por un voltaje de 7V, teniendo así una corriente máxima de 1 A.
- **Puerto VIN**: Permite aplicar una fuente de alimentación externa en el rango de 12 a 6 V a la entrada del regulador de la tarjeta Arduino. En este caso, NO se cuenta con protección contra inversión de polaridad ni contra sobre corriente. En caso de aplicar voltaje directamente al pin VIN, no se debe aplicar simultáneamente un voltaje en el Jack. No se recomienda conectar cargas más grandes que 1000 mA ya que este podría dañarse.

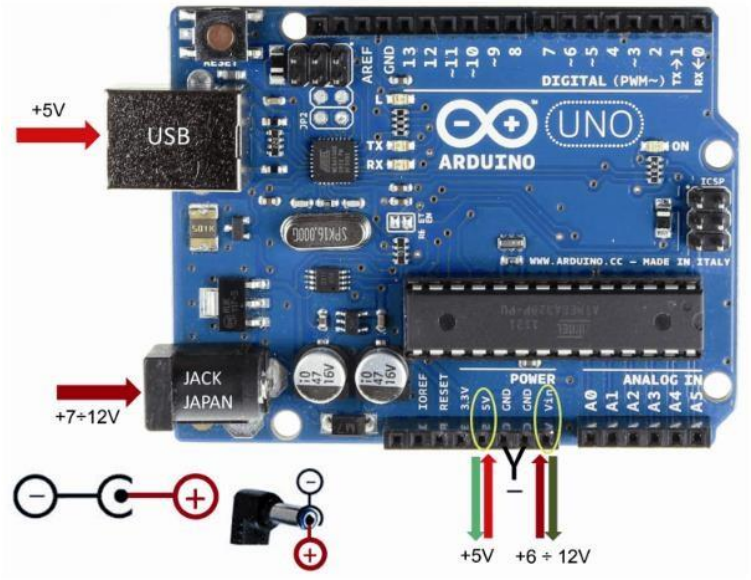



Figura 3.2.2.1: Tipo de Entradas de Alimentación en una placa Arduino UNO

La forma en la que se alimenta cualquier proyecto portable es con **baterías.** Lo que hay que ver es cuales son las más eficientes para este proyecto, y que detalles técnicos deberán poseer para funcionar correctamente:

- **Batería de plomo – acido (12 V),** alimentando el Arduino a través del *jack* de alimentación externa. ***Contra***: es el **peso y dimensión** de estas baterías, que podría no ir con la estructura del carro.
- **5 baterías Recargables AA NiMH** (En Serie -6 V a través de VIN)
    - ***Ventajas***: Son recargables, y son de gran duración.
    - ***Desventajas***: Son muchas baterías (aumenta el costo) recargarlas puede volverse tedioso. Densidad energética baja (1000 mAh)
- **4 baterías alcalinas** (En serie -6 V a través de VIN) podemos alimentar el Arduino a través del pin VIN.
    - ***Ventajas***: Son de bajo costo. Densidad energética media (1700 mAh)
    - ***Desventajas***: No son reutilizables.
- **Batería Cuadrada** (9V – *plug* conectado al jack de alimentación externa)
    - ***Ventajas:*** Fáciles de colocar y conseguir.
    - ***Desventajas***: No es eficiente, ya que tienen una vida útil muy corta y no poseen densidad energética elevada (500mAh) ni permite manejar intensidades altas.
- **2 baterías de Litio 18650**(3,7V a través de VIN)
    - ***Ventajas:*** Proporcionan alta capacidad de carga (hasta 4800 mAh), son recargables y solo se necesitan 2 para el funcionamiento del proyecto.
    - ***Desventajas***: Alto costo, necesidad de un cargador especial.
- **Arreglo de Baterías de Polímero de Litio 2 Celdas (Li-Po)** (7.4 V a través de VIN)

- ***Ventajas:*** Densidad de energía más alta (hasta 5000mAh), proporciona grandes intensidades (50A-100A), pequeño tamaño y peso.
- ***Desventajas***: Peligrosas de manipular, costo elevado.

Habiendo vistos las ventajas y desventajas de cada una de las opciones, elegimos implementar las **2 Baterías de Litio 18650**, presentándose como la opción más viable con relación a su costo-beneficio.

Figura 3.2.2.2: Baterías de Litio 18650

Sin embargo, destacamos que, para un proyecto de mayor envergadura y costo, una batería de tipo **Li-Po** podría ser algo a tener en cuenta, ya que poseen mayor densidad de carga, maneja mayores corrientes, y con un presupuesto más alto, podrían tomarse las medidas correspondientes de seguridad para su utilización.

Figura 3.2.2.4: Batería LiPo 5000 mAh

### 3.2.3. Programación

Para la programación del microcontrolador en el proyecto, utilizamos la IDE oficial de Arduino; un entorno de programación especialmente diseñado para programar este tipo de microcontroladores.

El lenguaje de programación es **C/C++**, pero más allá de que la lógica siga este lenguaje, existen diversas bibliotecas de Arduino en la que se requiere capacitación para aprender a utilizar, especialmente para lo que es la activación de pines, la lectura de ellos y el guardado de sus valores en variables.

En el **Anexo** se encuentra todo el código comentado en detalle para que se pueda como fue pensado el proyecto. Este mismo código, es el que está cargado en el microcontrolador actualmente.

Vamos a analizar por partes este código para entender en que consiste y como fue pensado, así se puede ver con facilidad cada una de las partes:

1. **Inclusión de Librerías**

```
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

Figura 3.2.3.1: Extracto de Código 1

Aquí se incluyeron las cuatro librerías utilizadas en el proyecto. Las **librerías** son conjuntos de implementaciones funcionales que permite el uso de todas esas, en el código en el que se la incluye. Esto permite utilizar libremente funciones definidas en las librerías, inclusive sus clases, tipos de variables y definiciones. Las incluidas incluyen respectivamente

- La librería **<SoftwareSerial.h>**, incluye todas las funciones necesarias para conectar las salidas de los pines de Arduino para conexiones seriales, distintas a la principal. Esta librería es la que permite configurar las conexiones por Bluetooth entre el dispositivo y el emisor.

- La librería **<Wire.h>** es utilizada para comunicarse con el protocolo I2C. Al incluirla, se incluyen funciones que permiten que no necesitemos una resistencia de *puy-up* a la salida del sensor. Permite el funcionamiento del sensor de Presión, Humedad y Temperatura.
- Las librerías **<Adafruit_Sensor.h>** y **<Adafruit_BME280>**, son dos librerías que permiten el funcionamiento de los sensores Adafruit, más específicamente, el sensor BME280, utilizado para medir Presión, Humedad y Temperatura.

## 2. Definiciones

Las definiciones son directivas en el lenguaje de programación que permiten definir **constantes.** Estas constantes son llamadas también como macros y permiten que al momento de escribir el nombre que defino, en ese lugar, se represente el valor numérico, cadena o expresiones elegidas.

```
//---De Motores---

#define IN1   45
#define IN2   44
#define IN3 43                          /
#define IN4 42                          /

//---De Sensores---

#define SEALEVELPRESSURE_HPA (1013.25)
#define GASSENSORPIN2 A0
#define GASSENSORPIN7 A1
#define LIGHTSENSORPIN A7

//--De Bluetooth---

#define BTRX  11
#define BTTX  10
```

Figura 3.2.3.2: Extracto de Código 2

Las constantes definidas **para los motores**, definen el número del pin que controlara las entradas del controlador de motores, dándole así el sentido y dirección a los dos motores utilizados.

Las constantes definidas **para los sensores**, definen el nivel del mar elegido (para cálculo de altitud), el pin analógico utilizado para controlar el sensor de Gas MQ-2 y para el MQ-7 y, por último, el pin analógico utilizado para controlar el sensor de luz.

Por último, las constantes definidas **para la conexión bluetooth** definen los números de pines para la recepción de datos del sensor (RX) y para él envió de datos del sensor (TX).

### 3. Variables Globales

Las variables globales son un tipo de variables, definidas antes de la función principal, que permiten ser usadas por todas las funciones en un código.

```
Adafruit_BME280 bme;
SoftwareSerial myBT(BTRX,BTTX);
char BTvalue;
```

Figura 3.2.3.3: Extracto de Código 3

La **primera** variable global, es una variable de tipo *Adafruit_BME280*; clase creada en una de las librerías que permite hacer, en tu BME280, las funciones de lectura de los parámetros ambientales solicitados.

La **segunda** variable global, es una variable de tipo *SoftwareSerial*; clase creada para definir un nuevo tipo de comunicación serial. "*myBT*" es el nombre elegido y entre paréntesis se eligen los pines elegidos para transmisión y recepción de datos.

La **tercera** variable global, es una variable de tipo *char* guarda los bytes enviados a través de bluetooth tanto para su lectura en el programa, y así, interpretar esas instrucciones y trasladarla a acciones específicas.

### 4. Declaración de Funciones

La declaración de la función o prototipo muestra está al compilador, permitiendo su utilización en el resto del código. No es lo mismo que la definición de la función; en esta etapa, se asocia un nombre con un tipo de dato y proporciona al compilador gran parte de la información de esta.

```
void Movement();
void EnviromentalStation();
```

Figura 3.2.3.4: Extracto de Código 4

Estas dos funciones van a ser las utilizadas a través del script, sin contar las incluidas en las librerías; una gestiona el movimiento del auto y la otra maneja la interpretación y envío de los datos del sensor, a la interfaz de usuario.

### 5. ***Setup* de Inicialización**

Antes de iniciar cualquier programa, necesitamos inicializar todas las condiciones necesarias para el funcionamiento del programa. Esta inicialización se realiza en una función que viene por defecto en el IDE, la función **voidsetup ()**. Esta función es leída solo una vez cada vez que se ejecuta el código. Sirve principalmente para la inicialización de los pines.

```
void setup(){
  myBT.begin(38400);
  Serial.begin(9600);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);


  pinMode(GASSENSORPIN2,INPUT);
  pinMode(GASSENSORPIN7,INPUT);
  pinMode(LIGHTSENSORPIN, INPUT);


  if (!bme.begin(0x76)) {

    Serial.println("NOT VALID!");
    while (1);
  }

}
```

Figura 3.2.3.5: Extracto de Código 5

- Las primeras **dos** inicializaciones, son las del Serial; una para el dispositivo bluetooth y otra para el Monitor Serial de IDE. Los valores entre paréntesis indican la velocidad de transmisión de datos, definida en bits por segundo (baudios). Es importante que

para cada dispositivo chequeemos las velocidades permitidas de transmisión de datos para su correcto funcionamiento.

- Las siguientes **cuatro** inicializaciones, son las que definen que los pines utilizados en el controlador de motores, serán usados como *OUTPUT* o salida, ya que los motores son actuadores y solo reciben datos.
- Las siguientes **tres** inicializaciones, son las que definen que los pines utilizados en cada uno de los sensores serán usados como *INPUT* o entrada, ya que los datos siempre llegan a la aplicación de los sensores, no se envían datos al sensor.
- Por último, inicializamos la comunicación I2C. Si no se puede inicializar el BME280 en la dirección física de este (0x76), se imprimirá en el monitor serial que no es válida y se quedara en un *loop* hasta que lo sea.

## 6. Función Principal

En todos los scripts realizados en Arduino, la función **voidloop ()**, es la función principal que ejecutara el programa. Se ejecutará repetidamente, hasta que el Arduino deje de ser alimentado. Cabe destacar que esta función, así como la de *setup*, vienen ya declaradas en el IDE por defecto, así que no requieren declaración en la etapa de Declaración de Funciones.

```
void loop()
{

      Movement();
      EnviromentalStation();
}
```

Figura 3.2.3.6: Extracto de Código 6

Decidimos en este proyecto utilizar solamente dos funciones en el código y ejecutarlas en *loop*. Esto permite visualizar bien cada una de ellas con sus descripciones y una mejor sectorización del código. Como mencionamos anteriormente, la primera es la función que controla el movimiento del automóvil y la segunda controla todos los sensores.

## 7. Funciones

### *a. Movement( )*

Esta función es la que se encarga de recibir del módulo Bluetooth, los movimientos indicados para que así, de las instrucciones correspondientes a los motores y estos puedan moverse en la dirección deseada. Esto lo realiza mediante la recepción de "*chars*" diferentes, cada vez que se manda la señal desde la APP para que se mueva en cierta dirección el robot.

```
if( myBT.available())
                {



                                BTvalue=myBT.read();

                                if(BTvalue=='A')
                                  {
                                   digitalWrite(IN1, HIGH);
                                   digitalWrite(IN2, LOW);
                                   digitalWrite(IN3, HIGH);
                                   digitalWrite(IN4, LOW);

                                  }
                                if(BTvalue=='B')
                                {
                                 digitalWrite(IN2, HIGH);
                                 digitalWrite(IN1, LOW);
                                 digitalWrite(IN4, HIGH);
                                  digitalWrite(IN3, LOW);
```

Figura 3.2.3.7: Extracto de Código 7

En esta parte del código, se ve encapsulada toda la función en un *if*, en el cual se le pregunta a la variable *myBT*, si se encuentra disponible para hacer la transmisión. Lo que hace aquí es chequear si los dispositivos están conectados por bluetooth. Eso se hace enteramente desde la interfaz de usuario. Una vez adentro del *if* y con los dispositivos conectados, se guarda el valor de lectura que está siendo enviado por el usuario al presionar un botón, dentro de la variable *BTvalue*. Esta variable es la que es leída dentro de los subsiguientes *if*.

- Al recibir la **letra A**, se envía al controlador de Motores el valor de cada uno de los pines; configurados, en este caso, para ir **Adelante**.
- Al recibir la **letra B**, se envía al controlador de Motores el valor de cada uno de los pines; configurados, en este caso, para girar a la **Derecha**.

```
if(BTvalue=='C')
 {
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);

 }
if(BTvalue=='D')
 {
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);

 }

if(BTvalue=='0')
{
 digitalWrite(IN1, LOW); // IN3 a 0
 digitalWrite(IN2, LOW);
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, LOW);
}
```

Figura 3.2.3.8: Extracto de Código 8

- Al recibir la **letra C**, se envía al controlador de Motores el valor de cada uno de los pines; configurados, en este caso, para girar a la **Izquierda**.
- Al recibir la **letra D**, se envía al controlador de Motores el valor de cada uno de los pines; configurados, en este caso, para girar hacia **Atrás**.
- Al recibir **la letra 0**, se envía al controlador de Motores el valor de cada uno de los pines configurados, en este caso, para **Frenarse.** En el caso de este proyecto, no existe botón de frenado. Esta letra se envía en todo momento en el que NO se está apretando ningún botón. Esto permite que el presionado de botón no sea continuo, sino que permita el movimiento solo mientras el botón está siendo presionado.

### *b. Enviromental Station ()*

Esta función es la que se encarga de enviar, a través del módulo Bluetooth, los valores de los sensores conectados a Arduino, para que esta información pueda estar disponible para el usuario. También realiza los cálculos necesarios para poder interpretar esa información de la forma correspondiente y en la unidad correspondiente.

```
float lreading = analogRead(LIGHTSENSORPIN);
float l_ratio = lreading / 1023.0;
l_ratio = pow(l_ratio, 2.0);


int   adc_MQ2 = analogRead(GASSENSORPIN2);
float mq2voltage = adc_MQ2 * (5.0 / 1023.0);
float Rs2=1000*((5-mq2voltage)/mq2voltage);
double GCgml=644.83*pow(Rs2/5463,-2.014)/1000;

int   adc_MQ7 = analogRead(GASSENSORPIN7);
float mq7voltage = adc_MQ7 * (5.0 / 1023.0);
float Rs7=1000*((5-mq7voltage)/mq7voltage);
double COgml=95.292*pow(Rs7/0.75,-1.556)/1000;
```

Figura 3.2.3.9: Extracto de Código 9

En esta primera parte de la función, se realizan los cálculos y conversiones entre el valor detectado y el valor que queremos sea visto por el usuario. Particularmente, esto corresponde a los pines analógicos:

Primero se leen los valores del pin donde está conectado el **TEMT6000** (Luminosidad). Una vez leído, se divide por el valor máximo posible (1023) para sacar el porcentaje y se eleva al cuadrado para un resultado más preciso

Lo siguiente es más complicado, y es la parte relacionada a los sensores de gases **MQ-2** y el **MQ7**. Para estos sensores, requerimos que los valores no fueran si había o si no había cierto gas, sino que nos dejara la concentración. El problema es que la lectura analógica y el valor real, no tienen una relación lineal y, encima, posee cierta sensibilidad a distintos tipos de gases. Por esta razón, es que necesitamos estimar la curva basada en los gráficos de sensibilidad provistos por el datasheet.

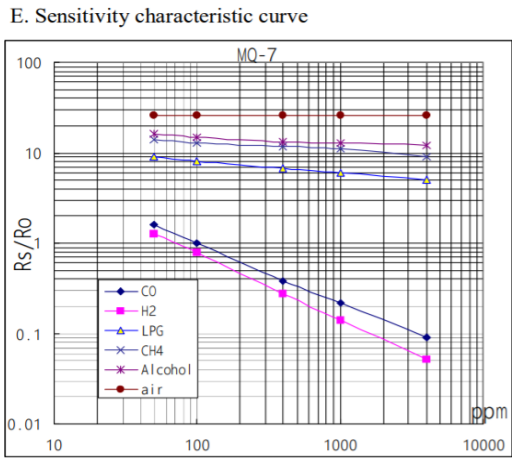


Figura 3.2.3.10: Curva Característica de Sensibilidad sensor MQ-7

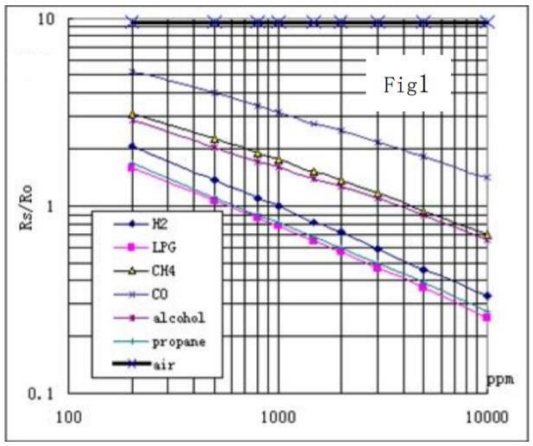



Figura 3.2.3.11: Curva Característica de Sensibilidad sensor MQ-2

Basados en esta curva y en el tipo de gas que queremos medir, hacemos una curva de regresión en Excel, basada en los valores de cada curva (el eje Y, posee la relación entre las resistencias Rs/Ro y el eje X las ppm o partes por millones).

TABLA VIII: Tabla de valores para Regresión MQ-7 (Monóxido de Carbono)

| MQ-7 -CO | |
|---|---|
| Rs/Ro | Ppm |
| 1,7 | 40 |
| 1 | 100 |
| 0,4 | 400 |
| 0,22 | 1000 |
| 0,09 | 4000 |

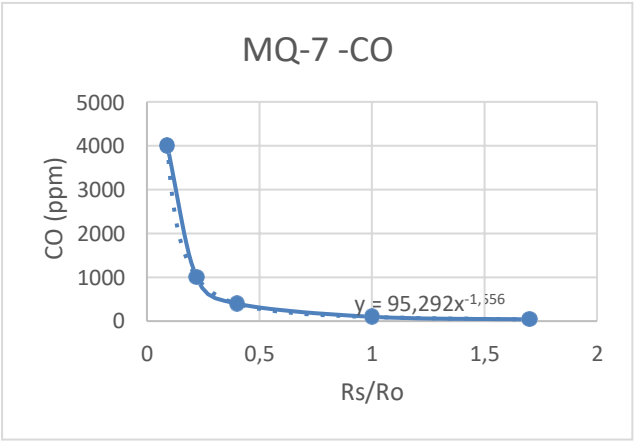



Figura 3.2.3.12: Curva de Regresión MQ7

TABLA IX: Tabla de valores para Regresión MQ-2 (Gases Combustible)

| MQ-2 - Gases Combustibles | |
|---|---|
| Rs/Ro | Ppm |
| 1,8 | 200 |
| 1,1 | 500 |
| 0,89 | 800 |
| 0,8 | 1000 |
| 0,65 | 1500 |
| 0,6 | 2000 |
| 0,48 | 3000 |
| 0,36 | 5000 |
| 0,25 | 10000 |

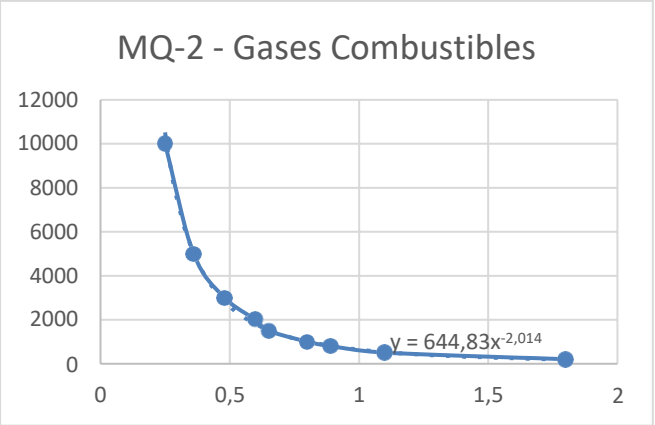



Figura 3.2.3.13: Curva de Regresión MQ2

De estos gráficos, saco las ecuaciones de regresión en donde X (Rs/Ro), es la variable independiente en la regresión. El valor de **Ro**es una constante que equivale al valor de la resistencia del sensor cuando se lo expone a una concentración conocida de aire ambiente (calculado en laboratorio) y **Rs** es la resistencia del sensor, la cual leemos desde el Arduino.

La información para el cálculo de Ro se saca de las Datasheet de los sensores y calibrando. Vemos que, en la parte superior de los dos gráficos de sensibilidad, hay una línea recta. Esa línea implica una relación lineal entre los parámetros Rs y Ro dada por un valor numérico.

Para el MQ-7:

$$\frac{R_{saire}}{R_o} = 28$$

Para el MQ-2:

$$\frac{R_{saire}}{R_o} = 9{,}8$$

Con estos valores, calculamos el valor numérico de Rsaire al calibrar los sensores y con eso obtenemos los valores de Ro. Para el MQ7 y MQ2, Ro es respectivamente 20 y 0,75

Ahora, lo único que nos queda es calcular el valor de Rs para reemplazar en la ecuación. Realizando una ecuación de divisor de tensión con el voltaje leído por el sensor, como se muestra en el código y suponiendo un valor de resistencia de carga RL de 1k, usando las siguientes ecuaciones:

$$R_s = 1000\ \Omega\ (\frac{5\text{V} - V_{MQ}}{V_{MQ}})$$

Una vez calculado Rs, ya podemos reemplazar su valor en la función dada por la regresión realizada en el gráfico, y determinamos la concentración del gas en **g/ml**.

$$ConcentracionCO = \frac{95{,}292}{1000} * (\frac{R_{s7}}{R_{o7}})^{-1{,}556}$$

$$ConcentracionGC = \frac{644{,}83}{1000} * (\frac{R_{s2}}{R_{o2}})^{-2{,}014}$$

Por último, en esta función, nos dedicamos a mandar por Bluetooth los valores de cada uno de los sensores, junto con su forma de medición.

```
    myBT.print(bme.readTemperature());
    myBT.print(" C");
    myBT.print("|");
    myBT.print(bme.readHumidity());
    myBT.print(" %");
    myBT.print("|");
    myBT.print(bme.readPressure());
    myBT.print(" hpa");
    myBT.print("|");
    myBT.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    myBT.print(" m");
    myBT.print("|");
    myBT.print((int)l_ratio);
    myBT.print(" %");
    myBT.print("|");
    myBT.print((int)COgml);
    myBT.print(" g/ml");
    myBT.print("|");
    myBT.print((int)GCgml);
    myBT.print(" g/ml");
    myBT.print("|");

return 0;
```

Figura 3.2.3.14: Extracto de Código 10

Al usar la función *print*, enviamos al dispositivo del usuario los valores dentro de los paréntesis. Enviamos en principio los valores de las lecturas de las funciones (las variables de clase BME, poseen funciones internas para leer los valores de los sensores y ponerlos en el lugar en la que son llamadas), las unidades que corresponden a las funciones y luego, una barra vertical. Esta barra, será leída por la interfaz de usuario como el método **para separar los distintos datos de los sensores.**

Esto será visto en la próxima sección, donde veremos a fondo cual será la interfaz de usuario y el programa que recibirá los datos de los sensores y que enviará los comandos de movimiento para el auto.

### 3.2.4. Interfaz de Usuario

Durante la realización de nuestro proyecto, nuestra idea fue que este proyecto pudiera ser utilizado en cualquier lugar y momento, de una manera fácil e interactiva. Existen diversos tipos de formas de comunicación entre el usuario y la maquina: podría ser mediante un control remoto, un ordenador u otros dispositivos digitales o analógicos.
Nuestra elección fue que el usuario pudiera comunicarse con este dispositivo a través de un artefacto que toda persona hoy en día lleva consigo a todo lugar y en todo momento**:** el teléfono móvil.
Hoy en día, los *smartphones* representan la mayor parte de la telefonía móvil a nivel global y poseen muchas prestaciones. Principalmente, los teléfonos móviles poseen dos sistemas operativos: Android o IOS (Apple).

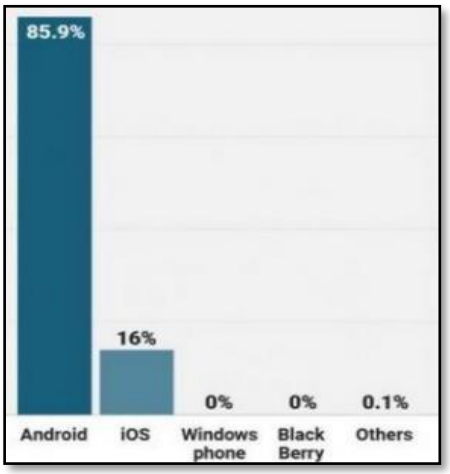



Figura 3.2.4.1: Porcentaje de uso de Sistemas Operativos en Smartphones (2017)

Podemos ver en el gráfico (Figura 7.2.4.1), como el sistema operativo más utilizado hoy en día es por gran diferencia Android. La ventaja principal de Android es que permite grandes libertades que IOS no. La más llamativa, es que permite instalar aplicaciones de terceros que no se encuentran en la tienda.

Esto permite a programadores independientes crear sus propias aplicaciones y utilizarlas libremente en su dispositivo móvil.

Por eso, elegimos que la interfaz con el usuario sea a través de una **Aplicación de Android (formato .apk)**.

La ventaja de las aplicaciones por sobre otros métodos de comunicación con el usuario es que:

- Se pueden descargar fácilmente y ser instaladas en pocos pasos.
- El teléfono móvil suele estar cerca de las personas constantemente, permitiendo que siempre puedan usar el auto.
- Interfaz visual de muy rápido entendimiento.
- Posibilidad de controlar flujos de datos de una manera sencilla.

Existen diversos métodos para la creación de aplicaciones que contienen sus diferentes grados de complejidades y de limitaciones. Por nuestro lado, nosotros queríamos crear una aplicación sencilla que pudiera controlar el auto a la vez de recibir datos de los sensores.

Nuestro método elegido fue la utilización de **MIT App Inventor II**. Esta página web fue creada como un entorno de desarrollo de software por GoogleLabs, MIT Media Lab y MIT

para crear aplicaciones para el sistema operativo Android. El usuario puede, a través de varias herramientas y de una forma visual, realizar aplicaciones simples de una manera interactiva.


Figura 3.2.4.2: Logotipo de MIT App Inventor

Si bien sirve para la creación de aplicaciones simples, este entorno posee diferentes ventajas:

- Requiere menos capacitación que otros entornos de desarrollo.
- Programación por Bloques, orientada a objetos directamente.
- Posee un simulador online para testear su funcionamiento.
- No es necesaria su descarga; se utiliza online.
- Guarda todos los archivos en la nube bajo tu nombre de usuario.

El entorno posee dos partes para su programación:

- **Solapa "Designer**": Aquí se realiza el diseño de las diferentes pantallas y se van creando los objetos como botones, títulos, nombres, tipos de conexiones, arreglos, etc. Se hace la parte de programación *front-end*.
- **Solapa "Blocks":** Aquí se realiza la programación por bloques. Es un tipo de programación no secuencial, que funciona a base de bloques que hacen que los objetos creados (en la solapa designer) realicen acciones e interactúen con el usuario.

Veremos estas dos solapas por separado, para explicar su funcionamiento básico. Pero primero mostraremos las pantallas del programa para entender que ve el usuario con anterioridad.

#### 3.2.4.1. Visualización

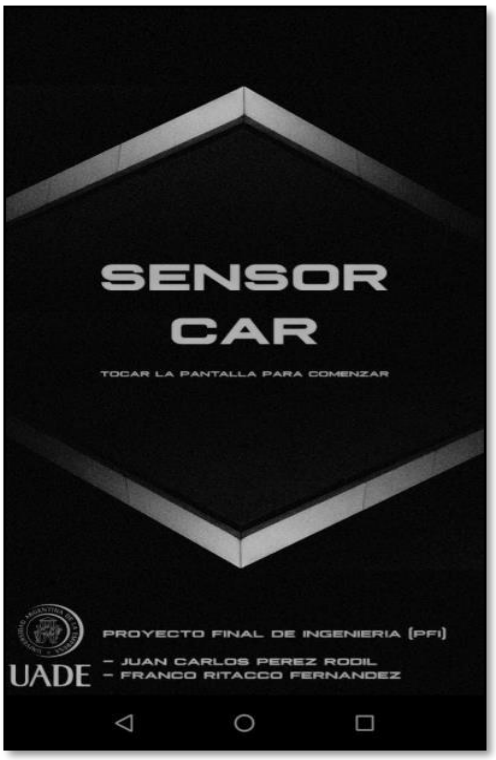



Figura 3.2.4.1.1: Pantalla de Inicio – Sensor Car App

Esta pantalla de inicio (ver Figura 3.2.4.1.1) es lo primero que ve el usuario al abrir la aplicación. Es una imagen diseñada por nosotros en la que aparece el título de la aplicación a la cual llamamos **Sensor Car** y abajo, los autores del proyecto. Una vez se presione la pantalla, como es indicado, se abrirá la parte principal aplicación. Cabe aclarar, que el manejo es táctil, como la mayor parte de los *Smartphones* hoy en día.

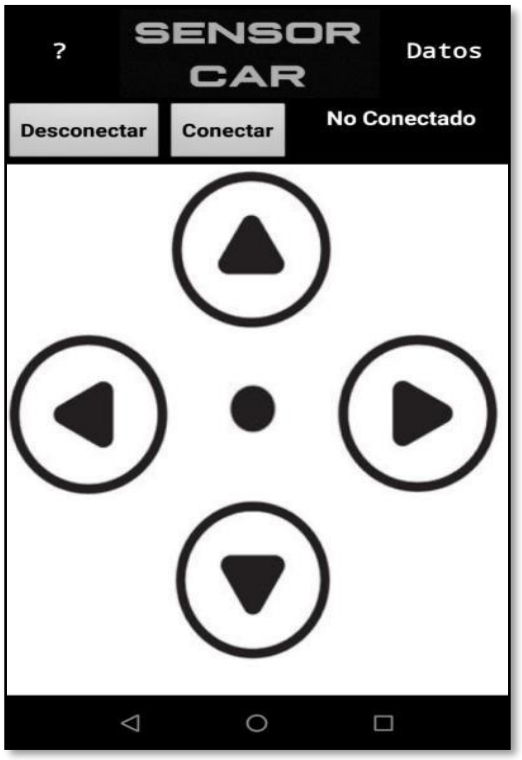



Figura 3.2.4.1.2: Pantalla de Movimiento (Desconectada)– Sensor Car App

En la siguiente pantalla (Figura 3.2.4.1.2) vemos las opciones que utilizaremos para el movimiento del auto y la conexión de este a Bluetooth. Primero, se debe presionar el botón conectar y saldrá una lista de las conexiones Bluetooth del teléfono. Previamente, por supuesto, hay que habilitar la conexión Bluetooth en teléfono. Allí, se debe seleccionar la opción **MI BT**, que es la correspondiente al módulo HC-05. Momentos después, aparecerá en la parte superior un cartel que dice “Conectado”, junto a un logo de Bluetooth.

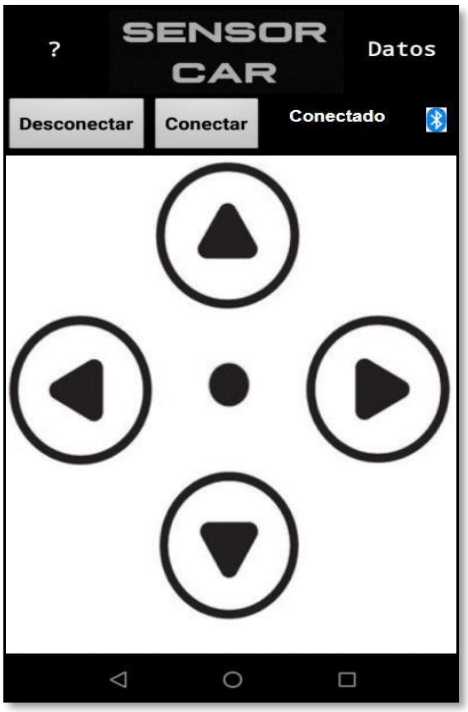



Figura 3.2.4.1.3: Pantalla de Movimiento (Conectada)– Sensor Car App

Cada flecha muestra una dirección de movimiento del Sensor Car y al presionarse, este debería avanzar, retroceder, rotar a la derecha o rotar a la izquierda. El punto del medio es decorativo.

Figura 3.2.4.1.4: Pantalla de Datos (Desconectada)– Sensor Car App

En esta pantalla se pueden ver todos los datos a recopilar por los sensores, que son enviados para su visualización en la aplicación. En caso de que no esté conectado a Bluetooth, o los sensores no estén recopilando bien los datos aparecerá como "Sin Datos". Una vez hayamos visto los datos necesarios, volvemos a la ventana anterior con el botón "Back".
La siguiente solapa es la que se titula "?" (ver Figura 3.2.4.1.3). En ella encontraremos los instructivos para la utilización de la aplicación y de nuevo, el botón de "Back".


Figura 3.2.4.1.5: Pantalla de Instructivos– Sensor Car App

### 3.2.4.2. Solapa "*Designer*"

Como explicamos anteriormente, en esta solapa se realizan todas las partes de diseño de la aplicación, así como la implementación de botones, elementos multimedia, carteles, comunicaciones y otros componentes programables y visibles. Explicaremos parte por parte, las secciones que posee la solapa Designer.

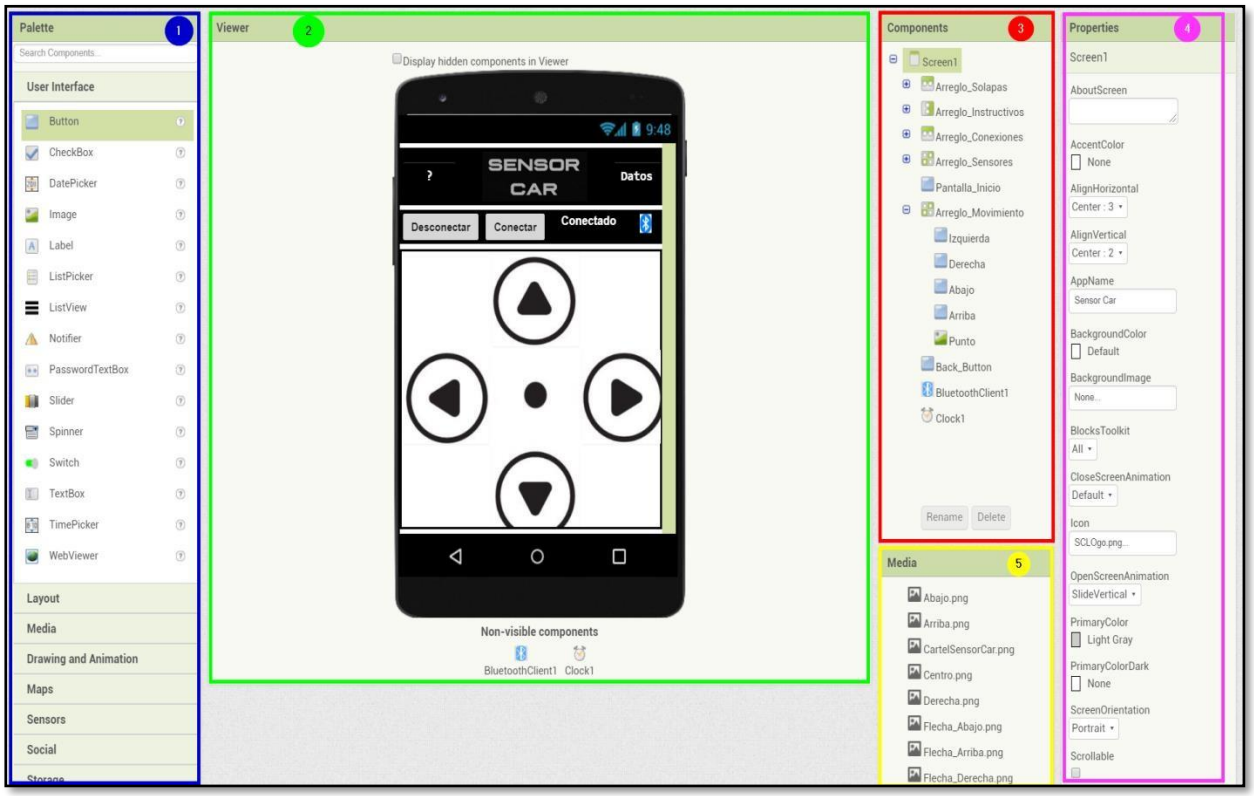



Figura 3.2.4.2.1: Layout Solapa “Designer”– MIT App Inventor 2

1) ***Palette***: La sección de *Pallete*, permite al usuario ver las distintas opciones de componentes que podrán incluirse en el programa. Estas deben ser arrastradas a la sección *Viewer* en la ubicación deseada, y una vez allí, aparecerán en el listado de componentes usados. Existen distintas categorías dentro de *Palette*, pero las más utilizadas por nosotros fueron:
    a. ***User Interface***: Es la más utilizada, y posee los componentes más comunes que interactúan con el usuario como botones, *checkbox*, listas, imágenes, careles, *switchs*, cajas de texto, entre otras.
    b. ***Layout*:** Permite la creación de **Arreglos**, tanto Verticales, Horizontales o Matrices, en donde se colocarán los componentes elegidos de la primera sección. Los arreglos permiten ubicarlos en distintas posiciones, agruparlos y configurarlos o programarlos en conjunto, entre otras opciones.
    c. ***Conectivity*:** Permite configurar para que la aplicación trabaje con cierto tipo de conexiones, ya sea Web o, como en este caso, Bluetooth.

d. ***Sensors:*** Permite configurar distintos sensores como acelerómetros, sensor de ubicación y poner relojes.

2) ***Viewer***: La sección de *Viewer* es en la cual veremos el diseño y disposición de los elementos en el programa. Así es como se vera la aplicación desde el teléfono móvil. En esta ventana solo se verán los elementos que estén tildados con la característica de "Visibles", salvo el caso en el que apretemos el botón "*Display hidden components in Viewer".* Ya veremos próximamente, porque esto es importante.

3) ***Components***: En esta sección figuran todos los componentes sacados de *Palette* y ubicados en *Viewer*. A estos se los puede renombrar y eliminar con los botones en la parte inferior, y seleccionar haciendo clic. Se ubican en forma de Árbol y aquí es donde es importante la utilización de los previamente mencionados Arreglos, ya que permiten organizar las diferentes secciones del trabajo. Como vemos, todos los componentes salen del Screen1. Esto es debido a que, en vez de crear diferentes pantallas, por una cuestión de simplicidad, se decidió **invisibilizar los arreglos que no están siendo utilizados en esa solapa.** Esto será visto en la solapa de "*Blocks*".


Figura 3.2.4.2.2: Listado Componentes (Arreglo Movimiento)– Sensor Car App

4) ***Properties***: Una vez seleccionamos un componente de entre la lista creada, se abrirá esta sección del lado derecho. Allí, se nos permitirá modificar distintos parámetros como ubicación, fuente, colores, tamaño, etc. Estos parámetros dependerán del tipo de componente seleccionado. Aquí también se podrá configurar la visibilidad en el *Viewer* del componente elegido. En la Figura 3.2.4.2.3, podemos ver como ejemplo, las propiedades del componente "Screen1".

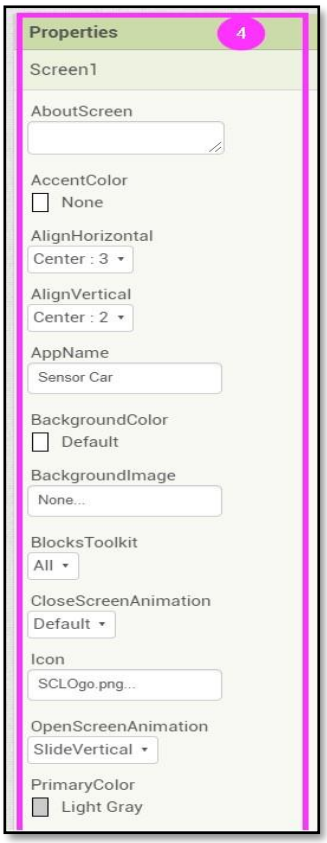



Figura 3.2.4.2.3: Propiedades de Screen1– Sensor Car App

5) ***Media:*** En este listado, aparecen todos los archivos multimedia subidos al proyecto y utilizados en componentes. En este proyecto, lo único subido aquí fueron imágenes.

Es importante entender que, en esta sección, se generarán todos los elementos que luego serán programados. Por eso, se debe tener claro que componentes agregar y cuáles serán sus funciones.

### 3.2.4.3. Solapa "*Blocks*"

En esta solapa del MIT Inventor 2, se programarán las conexiones lógicas entre los componentes, formando de esta forma el "código" *back-end.*

La forma de **programación en bloques** es la que utiliza MIT Inventor 2, siendo un método sencillo, muchas veces utilizado pedagógicamente. No se necesita saber ningún lenguaje de programación previo, simplemente se realizan las operaciones lógicas entre los distintos elementos, siendo esto un método mucho más visual que la escritura de código.

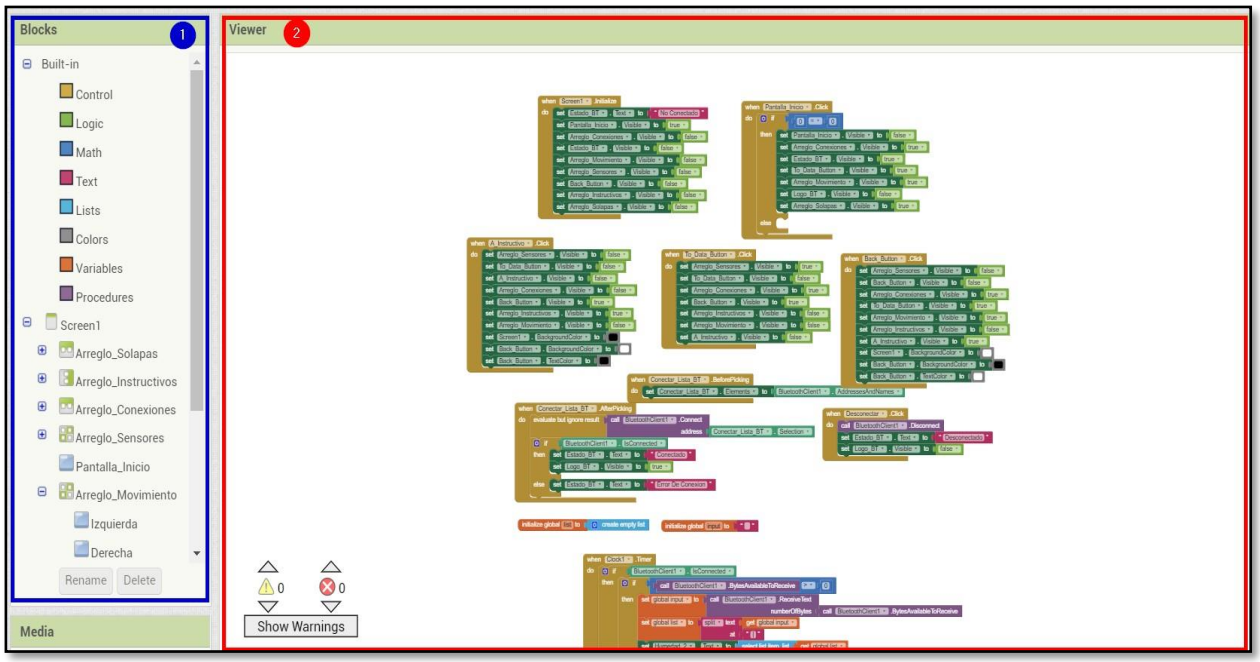



Figura 3.2.4.3.1: *Layout* Solapa "Blocks"– MIT App Inventor 2

En esta solapa solo existen dos secciones:

1) ***Blocks:*** En esta sección, se encuentran muchos bloques específicos del programa que realizan conexiones lógicas entre los distintos elementos. Hay dos categorías.
    a. ***Built-In:*** Bloques que ya están creados por defecto que poseen funciones o variables generales que son independientes de los componentes del programa. Existen categorías de Control (*ifs*, *while*, *when*), Lógicas (*true*, *false*, *equal*, *and*, *or)*, Matemáticas (senos, tangentes, números aleatorios), de Texto (reemplazar, subrayar, separar, etc.), de Listas (crear listas, comparar, asignar valores, longitud, etc.), Colores, Variables (iniciar, setear, conseguir, etc.) y, por último, Procedimientos.

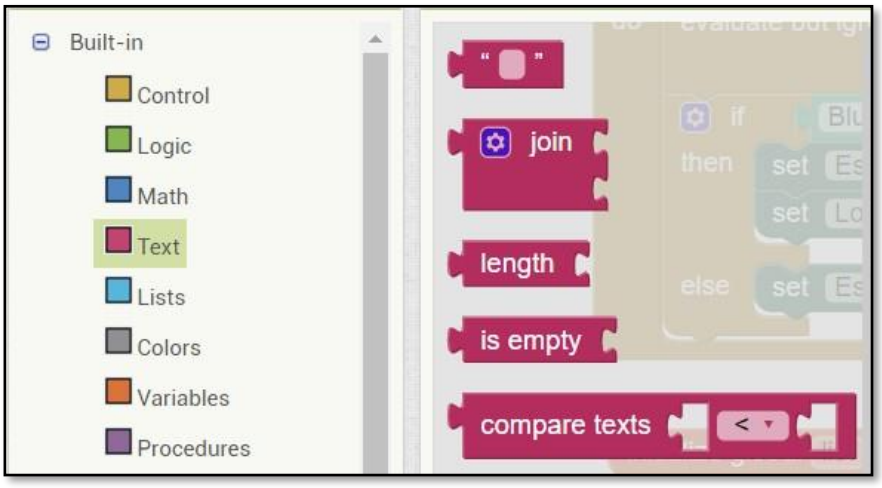



Figura 3.2.4.3.2: Extracto de Bloques *Built-In*: *Text*– MIT App Inventor 2

**b.** **Componentes:** En esta parte se puede seleccionar cada componente, y surgirán todas las opciones de bloques que existe para ese componente en particular. Cada tipo de componente tendrá opciones diferentes.

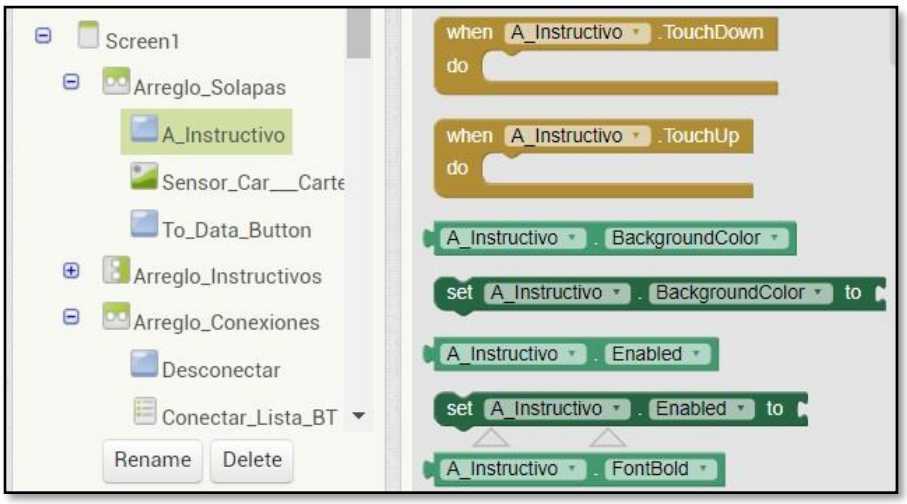



Figura 3.2.4.3.3: Extracto de Bloques de Componentes en Botón– MIT App Inventor 2

**2)** ***Viewer:*** En el *Viewer* es donde se conectan todos los bloques de código para formar las distintas partes del código.

A continuación, vamos a mostrar cómo fue realizado el código poniendo los extractos más importantes de este:

## I. Inicialización

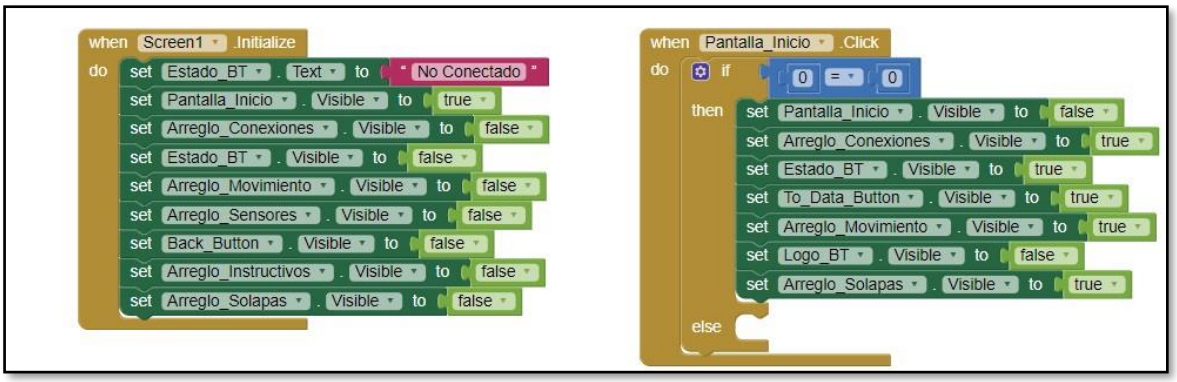



Figura 3.2.4.3.4: Bloques de Inicialización– MIT App Inventor 2

En esta etapa, inicializamos el programa a través el arranque de la "Screen1". Al mismo tiempo, seteamos todas las partes del programa que queremos que sean **visibles.** Lo que queremos ver en un inicio lo seteamos en invisibilidad como *true* y lo que no queremos ver como *false.* Al inicializar, lo único que queremos ver es la imagen **Pantalla_Inicio**.

En el siguiente paso, seleccionamos que cuando la imagen Pantalla_Inicio sea clickeada, se hagan visibles los arreglos que conforman la pantalla principal de nuestro programa (Arreglo_Conexiones, Estado_BT, Arreglo_Solapas, Arreglo_Movimiento, entre otras)

## II. Botones de Solapas

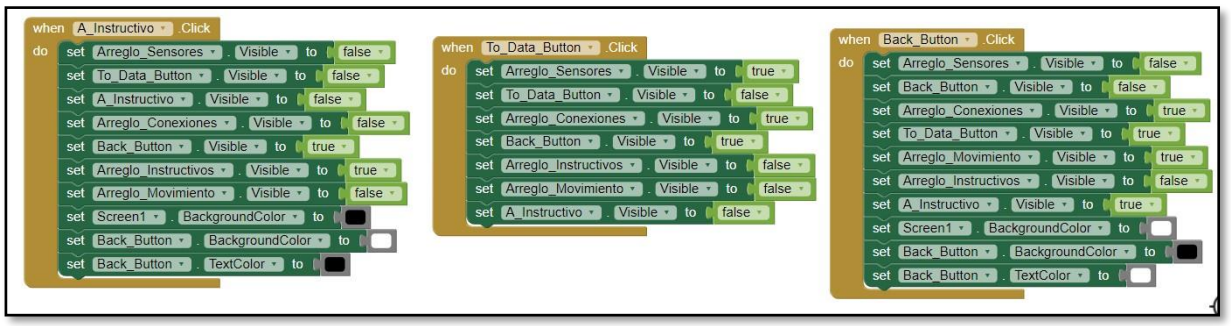



Figura 3.2.4.3.5: Bloques de Solapas– MIT App Inventor 2

Esta parte de bloques son las que manejan la visibilidad de las tres pantallas. Como explicamos antes, aquí definimos que cuando se presione el botón para acceder a alguna de las solapas del programa, el programa invisibilice todo lo que no corresponde a esa solapa, y

habilite (en las solapas A_instructivo y To_Data_Button) la informacióny el botón *Back*, el cual permite volver a la solapa de movimiento del Sensor Car.

## III. Botones de Conexión

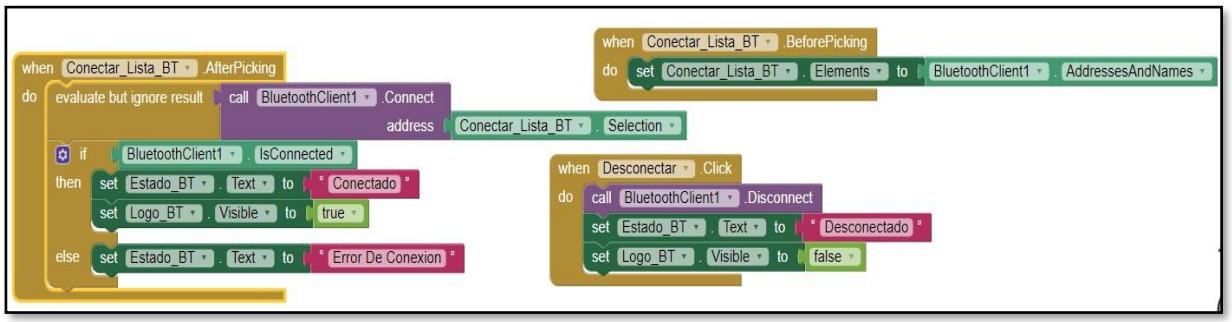



Figura 3.2.4.3.6: BloquesdeConexion– MIT App Inventor 2

En estos bloques se programa la conexión a Bluetooth. Al presionar el botón **Conectar**, llamamos al cliente de Bluetooth (previamente agregado en la solapa *Designer)* y elegimos que nos permita seleccionar el dispositivo a conectar. Antes de elegir (. *Before Picking),* programamos para que aparezcan todas las direcciones y nombres de los dispositivos vinculados por Bluetooth. Luego de elegir (. *AfterPicking),* elegimos que nos conecte a nuestra selección previo y que chequee la conexión. Si la conexión fue satisfactoria, ponemos que en el cartel de Estado_BT, aparezca como “Conectado”. En caso de que no se haya podido realizar la conexión, figurara “Error de Conexión”.

Al presionar el botón **Desconectar**, en caso de querer hacerlo, podemos desconectarnos de Bluetooth y figurara el texto “Desconectado”.

## IV. Botones de Movimiento


Figura 3.2.4.3.7: Bloques de Movimiento– MIT App Inventor 2

La forma de comunicarnos con la aplicación, es a través del envió de letras recibidas como *char* por el programa. Al clickear alguna de las flechas, se ejecutará la parte de. TouchDown. Allí, se enviará una letra que el código de Arduino interpretará para darle el movimiento a los motores en la dirección deseada. Una vez se suelta el botón (. *TouchUp*), se enviará el *char* "0". Esto, el código lo interpretara como que los motores deben frenarse.

De esta forma, el Sensor Car solo se moverá mientras los botones estén apretados, y no perpetuará su movimiento luego de ser soltado. En algunos casos puede existir un leve delay, que viene por parte del envío de datos por Bluetooth.

## V. Recepción de datos de los Sensores

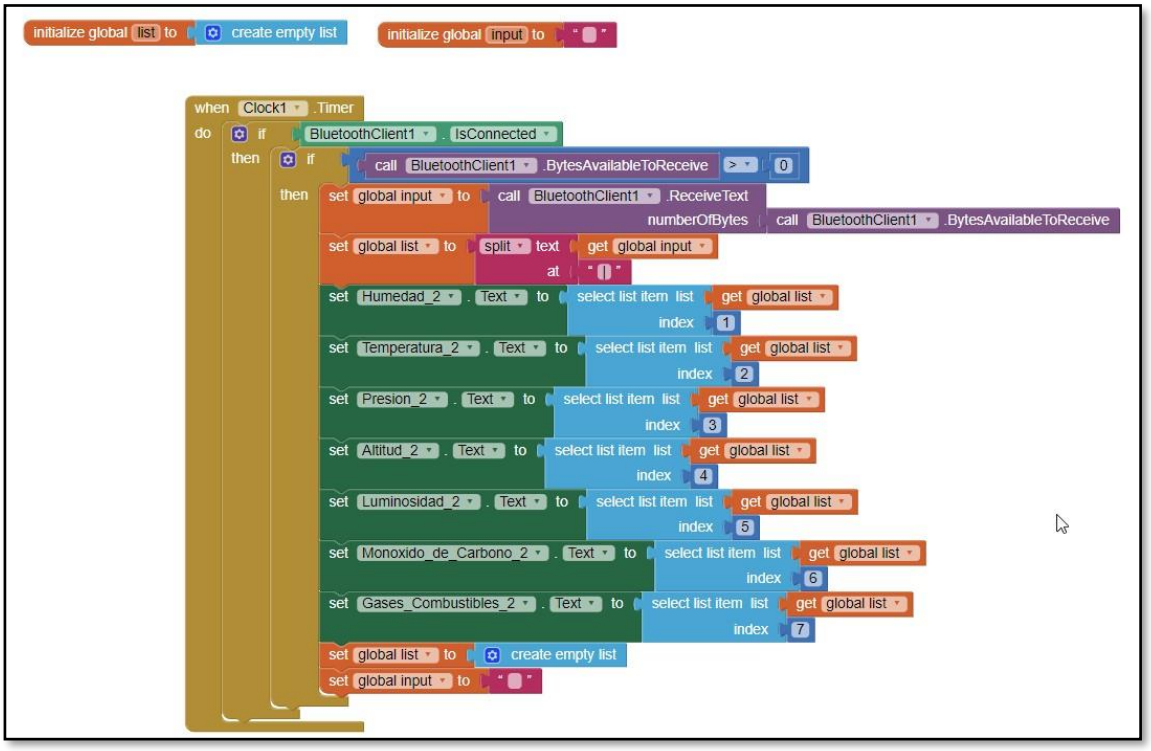



Figura 3.2.4.3.8: Bloques de datos de Sensores– MIT App Inventor 2

En este gran bloque, es donde se gestionarán todos los datos de todos los sensores. Los pasos para la lectura de la información de ellos son los siguientes:

1. Se crea una **lista** en donde se guardarán los distintos datos de los sensores. En esa lista, al principio solo habrá un cero, definida por el valor de **input**.
2. Se inicializa un temporizador con el reloj agregado anteriormente, que ejecutará la lectura de sensores a tiempo real.
3. En caso de que estemos conectados correctamente a Bluetooth, llamaremos al cliente para decirle que recibiremos ciertos *bytes* por este medio. Luego, se indicará que estos *bytes* serán todos de texto.
4. Seteamos para que cada *string* de datos recibido sea separado con una barra vertical, y guardado en a la lista creada anteriormente. Esto permitirá que cada información de cada sensor, sea separada y que la información enviada a través de los sensores sea segmentada.
5. Llamamos a todos los sensores, agarrando los datos de la lista en la que fueron guardados previamente, con el número de índice indicado (**Ej**: El sensor de Humedad

será el de índice 1, el de Temperatura el 2, etc.). En ese momento, se setea el texto de cada espacio en la solapa "Datos", para que presente el valor enviado por el Arduino. En este valor, se incluirá tanto el valor numérico, como las unidades.

Una vez terminemos la programación y el diseño en las dos solapas, crearemos el archivo APK, (ver Figura 3.2.4.3.9), pasando primero por una compilación. Lo deberemos abrir en el teléfono móvil para instalar la aplicación, así como hacemos con cualquier otra.

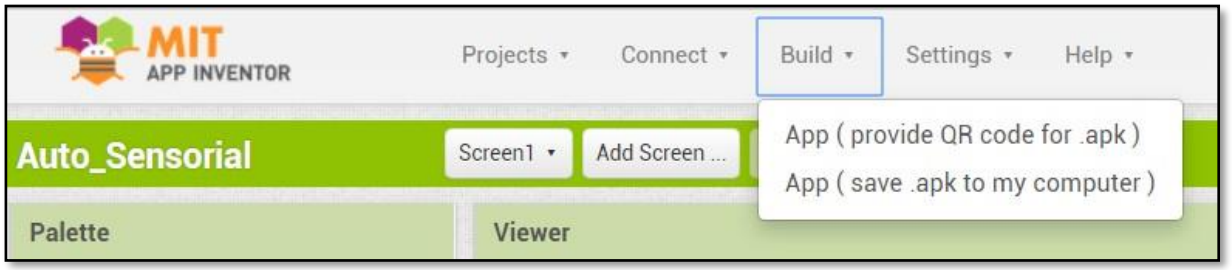



Figura 3.2.4.3.9: *Build*– MIT App Inventor 2

Esta aparecerá con el nombre "Sensor Car" y su logo, y ya podremos acceder a ella para su uso inmediato.

### 3.2.5. Diseño Estructural y Montaje

Para el montaje del robot, se comenzó con las ruedas y rodamientos. La estructura cuenta con 2 ruedas dentadas que son las que trasmiten el movimiento a las orugas y otras 4 ruedas que giran locas y sirven de guía para la transmisión. Las primeras 2 formadas cada una por los

TABLA X: Composición Física de Ruedas Dentadas – Sensor Car

| | Pieza | Cantidad |
|---|---|---|
| 1 | Espaciador hexagonal de cobre de 28 mm | 3 |
| 2 | Acoplamiento | 1 |
| 3 | Tornillo métrica 3, paso 8 | 6 |
| 4 | Conector | 1 |
| 5 | Tornillo métrica 2 | 1 |
| 6 | Tornillo extractor | 2 |
| 7 | Rueda dentada | 2 |

Primero se instalaron los tornillos de fijación en el acoplamiento. Asegurándose de que no sean demasiado profundos para permitir la inserción del motor más adelante. Luego se instalaron los espaciadores hexagonales de cobre de 28 mm en uno de los discos de rueda con los tornillos M3 * 8. Se instalaron los discos de la segunda rueda con los tornillos M3 * 8 restantes. Se le colocó el acoplamiento de aleación de aluminio en el conjunto de la rueda desde el lado del disco de la rueda que proporciona un gran agujero central en forma para el acoplamiento. Asegurándose de que los tornillos de fijación estén en el exterior de la rueda. Por último, se completó instalación de la rueda usando el tornillo M4 * 16 del otro lado del disco de la rueda motriz.

Figura 3.2.5.1: Chasis T100

Las otras 4 ruedas están formadas por:

TABLA XI: Composición Física de Ruedas NO Dentadas– Sensor Car

| | Pieza | Cantidad |
|---|---|---|
| 1 | Espaciador hexagonal de cobre de 17 mm | 3 |
| 2 | Tornillo métrica 3 paso 8 | 6 |
| 3 | Conector | 1 |
| 4 | Tortilla M2 | 1 |
| 5 | Rodamiento | 2 |
| 6 | Disco de Rueda | 2 |

Armadas de manera similar a las dentadas, primero se instalaron los 3 espaciadores hexagonales en un disco de rueda usando los tornillos M3 * 8 para sujetarlos en el disco de la rueda. Para el centro el conector con rodamiento, ajustado con el tornillo M2.

Figura 3.2.5.2: Chasis T100

Para el montaje de la base de aluminio, las ruedas, la oruga y los motores, se necesitaron las siguientes piezas:

TABLA XII: Composición Física Base– Sensor Car

| | Pieza | Cantidad |
|---|---|---|
| 1 | Motor 9v | 2 |
| 2 | Orugas | 2 |
| 3 | Paneles de aluminio (superior y laterales) | 4 |
| 4 | Tornillo M3 * 12 | 4 |
| 5 | Tuerca M3 * 8 | 4 |
| 6 | Tornillo M3 * 10 | 8 |

Primero se instalaron las ruedas a los paneles laterales, ambos con tornillos M3 * 8, también los motores de 9V con tornillos M3 * 10. Quedando como los de la figura XXX

Figura 3.2.5.3: Chasis T100

Luego, se colocaron los paneles centrales con tornillos M3 * 10, se instalaron las ruedas dentadas de transmisión con el conector instalado previamente. Ya con la estructura ensamblada se continuó con la colocación de la oruga. Se utilizó un destornillador pequeño para desarmarla teniendo en cuenta que la pista está conectada individualmente a través de una aguja y modificamos su longitud para buscar la tensión adecuada. Obteniendo así, el chasis completo del robot.

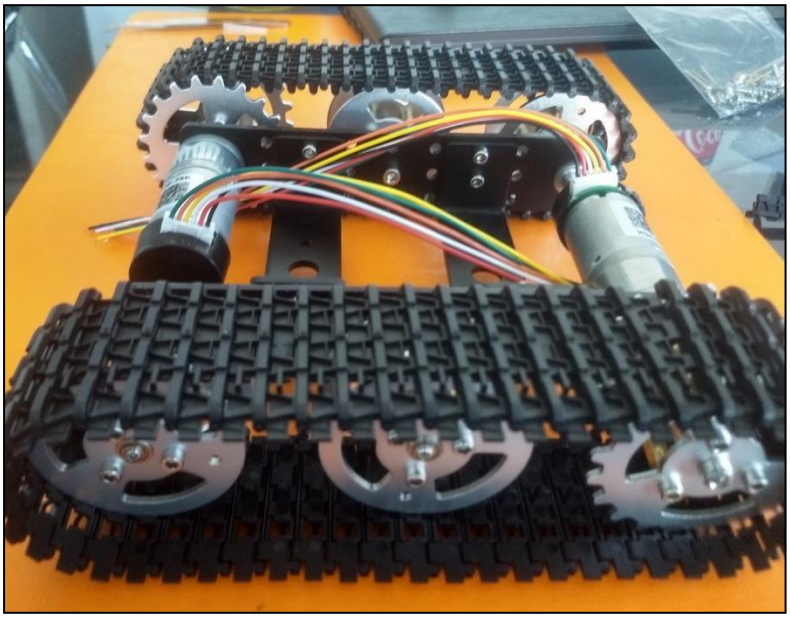


Figura 3.2.5.4: Chasis T100

Construimos el robot en CAD con el objetivo de una visión más clara, la posibilidad de escalabilidad y de realizar simulaciones. Para esto usamos la herramienta SolidWorks. Manteniendo las dimensiones originales, pudimos tomar diferentes capturas e incluso hacer la vista explosionada.

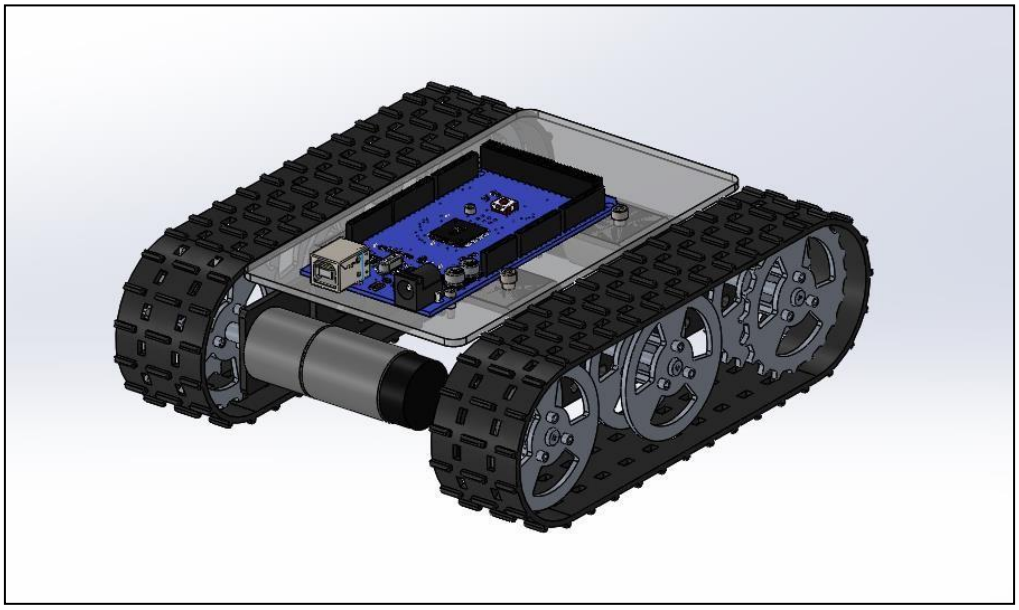


Figura 3.2.5.5: Ensamble Completo CAD – Sensor Car

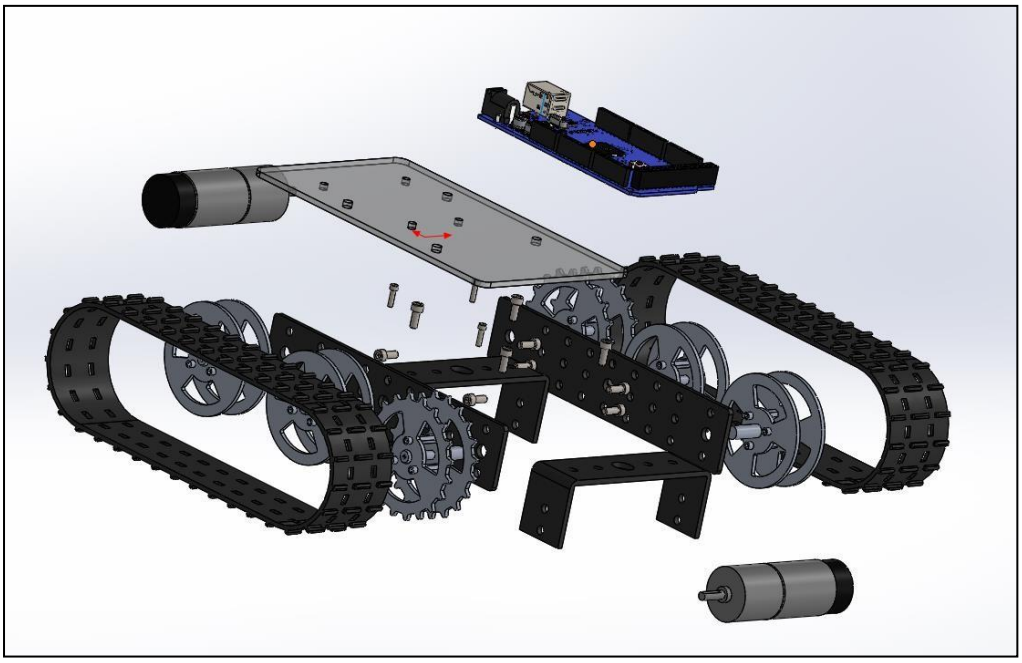


Figura 3.2.5.6: Ensamble Completo CAD (vista Explosionada) – Sensor Car

Realizamos una simulación por deformación con una carga de 5kg aplicada al acrílico. La estructura no sufrió deformaciones significativas y, además, con una masa 1000% mayor al de trabajo. Teniendo en cuenta para esto, que el microcontrolador con todos los sensores no

pesaba más de 0.5kg. La simulación se hizo tomando las siguientes propiedades de los materiales:

TABLA XIII: Hipótesis de Parámetros Físicos y Resistencia de Materiales– Sensor Car

| | Acrílico | | Aleación Aluminio 1100-H12 | |
|---|---|---|---|---|
| **Propiedad** | **Valor** | **Unidad** | **Valor** | **Unidad** |
| **Módulo de elasticidad** | **3000** | **N/mm^2** | **68899.9998** | **N/mm^2** |
| **Relación de Poisson** | **0.35** | **N/A** | **0.33** | **N/A** |
| **Módulo de corte** | **890** | **N/mm^2** | | **N/mm^2** |
| **Densidad de masa** | **1200** | **kg/m^3** | **2710** | **kg/m^3** |
| **Fuerza de Tensión** | **73** | **N/mm^2** | **109.99999** | **N/mm^2** |
| **Limite elástico** | **45** | **N/mm^2** | **99.284505** | **N/mm^2** |
| **Coeficiente de expansión térmica** | **5.20E-05** | **/K** | **2.40E-05** | **/K** |
| **Conductividad térmica** | **0.21** | **W/ (m.K)** | | **W/ (m.K)** |

Con la fuerza distribuida como muestran las flechas violetas y el punto de contacto con el suelo puesto en las ruedas como muestran las flechas verdes, tuvimos los siguientes resultados (ver Figura 3.2.5.7):

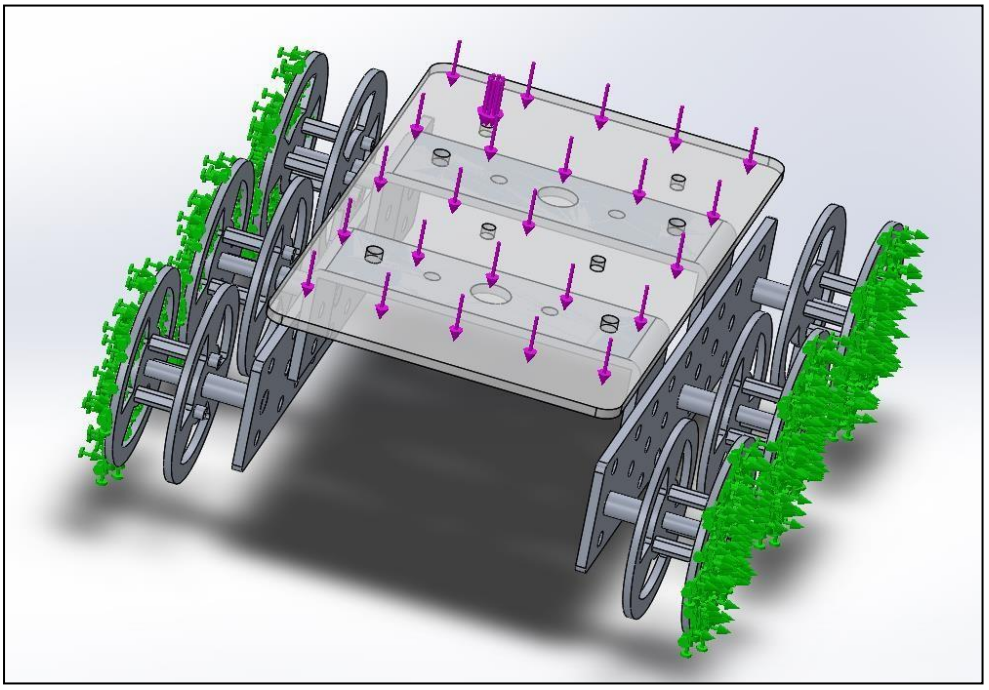


Figura 3.2.5.7: Cargas Aplicadas en Simulación – Sensor Car

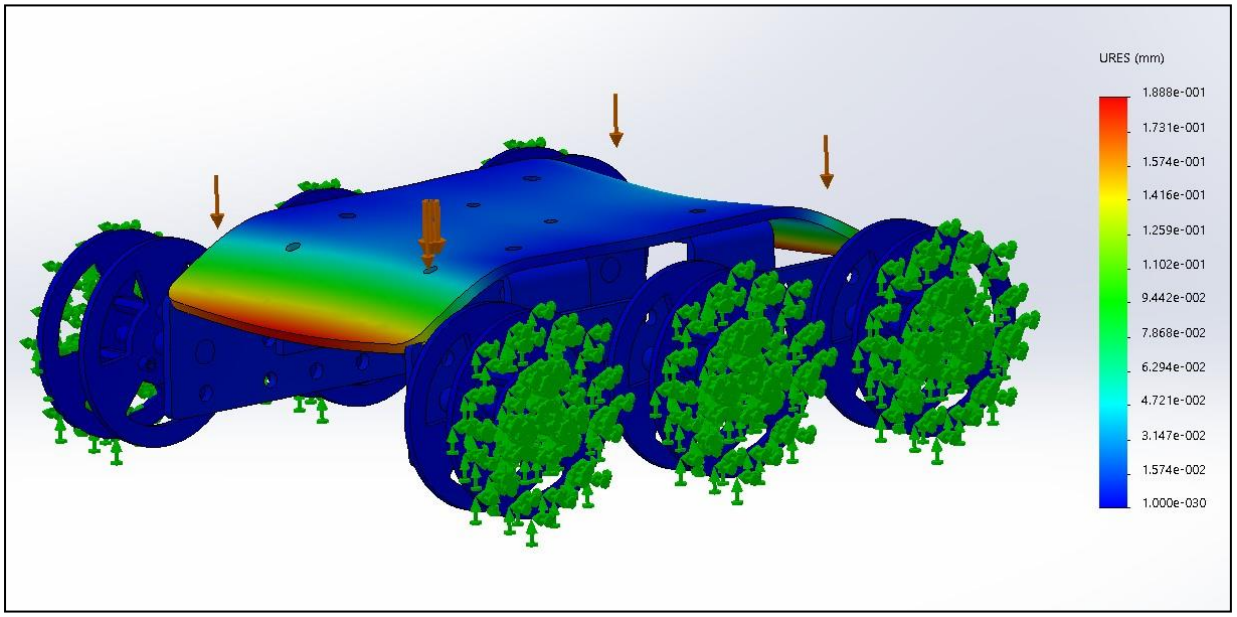



Figura 3.2.5.7: Simulación de Deformación (5kg de Carga Distribuida) – Sensor Car

Los resultados obtenidos en las simulaciones muestran pequeñas deformaciones. Del orden de 1.888e-001 mm en los bordes y 3.14e-002 mm en el centro de la placa de acrílico. Por otro lado, la estructura de aleación de aluminio básicamente no sufre deformaciones (1.000e-030 mm). Esto nos demostró que estructuralmente el robot era resistente para el uso que le íbamos a dar y que podría soportar la carga sin complicaciones.

## 3.3. Propuestas de Escalabilidad

A lo largo del proyecto, nos dimos cuenta que, en muchas ocasiones, existían varias alternativas a las elegidas que permitirían la escalabilidad del proyecto. Algunas de ellas no fueron elegidas en el proyecto por varios motivos, desde la complejidad en su uso, hasta el costo, siendo este ultima la principal razón. Mas allá de que no se hayan implementado y estuvieran fuera del alcance definido para el proyecto, consideramos productivo mencionar algunas de ellas y las limitaciones que encontramos para su incorporación.

**Conexión Wifi**

La implementación de Wifi aumenta muchísimo las posibilidades de uso del proyecto. Anteriormente, mencionamos que la conexión Bluetooth era la más adecuada para el proyecto, principalmente, por su simplicidad de uso para el usuario. Pero la realidad, es que se pueden tener **los dos protocolos de comunicación simultáneamente**, por supuesto, por un mayor costo y una mayor complejidad del código. Algunos beneficios de esta conexión adicional son:

- Posibilidad de conexión a redes de dispositivos.
- Posibilidad de control a distancias geográficamente distantes.
- Implementación de dispositivos que solo utilizan conexión Wifi.

Una de las maneras más sencillas para agregar es la mencionada anteriormente: incorporando un módulo con un chipset ESP8266.

**Cámara IP**

Una cámara IP siempre es una gran adición para todo tipo de proyectos, particularmente los que contienen un objeto en movimiento. Una cámara IP permite ver todo lo que está pasando a su alrededor, manejándola desde una dirección IP en la Web. Es muy sencilla de controlar, solo requiere acceso a un explorador web. Facilitaría la observación de espacios confinados, además de que no requiere que una persona este observando directamente al dispositivo mientras lo conduce. MIT App Inventor 2, permite incluso incluir una interfaz de explorador web y así se podría implementar perfectamente en la interfaz de usuario.


Figura 3.3.1: Cámara IP Giratoria

Sin embargo, algunas complicaciones que surgieron cuando pensamos en esta implementación fueron:

- Los altos costos de estas cámaras (salvo las que son módulos de Arduino, pero no permiten filmaciones en vivo).
- La necesidad de una conexión Wifi.
- La alimentación de la cámara debería ser muy grande para un proyecto móvil, reduciendo la duración de la batería o necesitando una más grande y pesada.

**Bluetooth 5 y LoRa**

El Bluetooth 5, es la quinta versión del protocolo de comunicación Bluetooth. Sin ir a detalles técnicos, esta nueva versión permite transferir datos dos veces más rápido, con un ancho de banda nueve veces mayor y, lo más importante, permite transferir a **distancias más de cuatro veces mayores a la versión anterior.** Además, permite transmitir conexiones a mas


Figura 3.3.2: Logotipo de Bluetooth 5

Este tipo de tecnología es muy novedosa, y empezó a surgir en estos últimos años. Particularmente, para incorporarla con Arduino existe un módulo**HM-18** con chipset CC2640R2F que permite este tipo de conexión.

Figura 3.3.3: Modulo HM-18

Las mayores limitaciones que encontramos para la incorporación de esta tecnología son:

- El costo del módulo, que es tres veces mayor al utilizado (U$D 12).
- La necesidad de un Smartphone que contenga esta tecnología, acotando el proyecto solo a gente con modelos superiores.

Otro posible protocolo de comunicación es **LoRa**, creado por Semtech y ampliamente utilizado en la industria de IoT (Internet of Things). Es una técnica de espectro modulado derivada de CSS que tiene largo alcance (hasta 12km) y bajo consumo de poder.


Figura 3.3.4: Logotipo de LoRa de Semtech

Para proyectos en los que se necesite un largo alcance, es el protocolo de comunicación ideal. El problema, es que hacerlo comunicar con una interfaz de usuario como una aplicación móvil conlleva una complejidad alta, además de los altos costos de su implementación.

# 4. Conclusión

En conclusión, consideramos que el proyecto nos permitió desarrollar una herramienta útil, económica y con posibilidad de utilización para los objetivos deseados. También se dejó una sólida base para su escalabilidad, proporcionando muchas herramientas de desarrollo altamente versátiles que permiten la incorporación de nuevas tecnologías más allá de las aquí presentadas. La faceta más interesante del proyecto es la cantidad de conceptos de diversas ramas de la Ingeniería que este integra y como todos se unen para dar lugar a un dispositivo de estas características.

Durante la realización, surgieron innumerables desafíos. En la parte de selección de componentes existieron problemas principalmente derivados de los altos precios de los productos y la imposibilidad de importación de muchos productos simultáneamente en Argentina. Esto dificulto mucho la contabilización de los costos del prototipo, ya que se tuvieron que implementar muchos supuestos para realizar las cuentas (tipo de cambio fijo, inflación mensual baja, etc.).

En la parte de implementación, debimos capacitarnos en diversas áreas que nos permitieron aplicar muchos conocimientos universitarios, en tecnologías prácticas y modernas, en las cuales debimos realizar cursos web para poder desarrollarlas. Surgieron complicaciones a la hora de realizar el prototipo, debido a problemas informáticos de niveles de complejidad no contemplados, pero se pudieron solucionar con diversas investigaciones y mediante muchas pruebas y errores.

Los objetivos del trabajo fueron cumplidos satisfactoriamente. El automóvil diseñado y construido, puede realizar todas las acciones que fueron propuestas en su alcance y es apto para escalabilidad.

# 5. Bibliografía


- "*Ultimate Guide For Arduino Sensors/Modules*", SimonKnight, 2018. ISBN-13: 9781730831539.

- "Sensores y Actuadores: Aplicaciones con Arduino", Leonel Corona Jimenez, Grupo Editorial Patria, 2014. ISBN: 9786074388008

- "Programación en C++ para Ingenieros", Fatos Xhafa, 2006.ISBN-13: 978-8497324854.

- "*Learning MIT App Inventor*", Derek Walter, Mark Sherman, 2014. ISBN-13:9780133798630.

- "Ingeniería de Control Moderna", Katsuhiko Ogata, PEARSON 2010. ISBN-13: 9788420536781

- *"*El Gran Libro de Solidworks*"*, Sergio Gomez, Marcombo 2007 ISBN:9788426714589

- *"*Resistencia de Materiales*",* James Gere, PARANINFO 2010. ISBN: 9788497320658

# 6. Referencias


- https://hetpro-store.com/TUTORIALES/microcontrolador/
- https://leantec.es/diferencias-entre-arduino-y-raspberry-pi/
- https://www.xataka.com/basics/arduino-raspberry-pi-que-cuales-sus-diferencias
- http://arduino.cl/arduino-nano/
- https://www.mecatronicalatam.com/tutorial/es/sensores
- https://blog.reparacion-vehiculos.es/a-que-temperatura-debe-estar-tu-taller-mecanico
- https://programarfacil.com/podcast/82-escoger-mejor-sensor-temperatura-arduino/
- https://www.luisllamas.es/arduino-dht11-dht22/
- https://naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html
- https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/
- https://www.intorobotics.com/common-budgeted-arduino-light-sensors/
- https://www.mecatronicalatam.com/resistencia/fotoresistor
- https://www.adslzone.net/2014/11/27/wifi-vs-bluetooth-diferencias-ventajas-e-inconvenientes/
- https://www.prometec.net/arduino-wifi/
- https://lastminuteengineers.com/bme280-arduino-tutorial/
- https://www.motioncontroltips.com/faq-what-are-hall-effect-sensors-and-what-is-their-role-in-dc-motors/
- https://www.luisllamas.es/arduino-i2c/
- https://learn.sparkfun.com/tutorials/temt6000-ambient-light-sensor-hookup-guide/all
- https://www.geekfactory.mx/tutoriales/tutoriales-arduino/alimentar-el-arduino-la-guia-definitiva/
- https://www.luisllamas.es/alimentar-arduino-baterias/
- https://appinventor.mit.edu/
- http://catedratelefonica.ulpgc.es/IXNAMIKI-OLINKI-el-robot-de-rescate
- https://es.gizmodo.com/colossus-asi-es-el-robot-de-los-bomberos-de-paris-que-1834102424
- https://www.sciencedirect.com/science/article/pii/S1697791214000788
- https://es.aliexpress.com/item/33042962542.html?spm=a2g0o.productlist.0.0.5f912d26yhkEcH&algo_pvid=7869d976-9326-414e-a637-41e6d9d9eef6&algo_expid=7869d976-9326-414e-a637-41e6d9d9eef6-0&btsid=3ef26182-30d0-4eeb-80cb-81a694600efd&ws_ab_test=searchweb0_0,searchweb201602_9,searchweb201603_52
- https://es.aliexpress.com/item/32876365731.html?spm=a2g0o.productlist.0.0.5f912d26yhkEcH&algo_pvid=7869d976-9326-414e-a637-41e6d9d9eef6&algo_expid=7869d976-9326-414e-a637-41e6d9d9eef6-4&btsid=3ef26182-30d0-4eeb-80cb-81a694600efd&ws_ab_test=searchweb0_0,searchweb201602_9,searchweb201603_52
- https://es.aliexpress.com/item/32893700845.html?spm=a2g0s.9042311.0.0.7b7563c0eszLbg

# ANEXO A: Código

```
1.  //                                      Proyecto Final de Ingeniería (PFI)
    ----------
2.  //
3.  //           Proyecto: "Desarrollo de un Robot AutomovilMultisensorial para la obtencion de
4.  //                     parámetros del Medio Ambiente circundante"
5.  //        Integrantes: Perez Rodil, Juan Carlos. Ritacco, Franco.
6.  //              Tutor: Zambrano, Daniel
7.
8.
9.
10. //           LIBRERIAS
11.
12.
13. #include<SoftwareSerial.h>               //Librería utilizada para realizar la entrada Serial,
    a traves del modulo HC-05 de Bluetooth
14. #include<Wire.h>                         //Librería utilizada para comunicarse a través de
    comunicacion I2C, sin necesidad de resistencias de pull-up
15. #include<Adafruit_Sensor.h>              //Librería con funciones para sensores con protocolo
    de comunicacion I2C
16. #include<Adafruit_BME280.h>              //Librería con funciones especificas para el sensor
    BME280 (Temperatura, Humedad y Presión)
17.
18.
19. //          DEFINICIONES
20.
21.
22. //---De Motores---
23.
24. #define IN1   45                        // IN1 de L298N a pin digital 45. Controla el motor 1
25. #define IN2   44                        // IN2 de L298N a pin digital 44. Controla el motor 1
26. #define IN3 43                       // IN3 de L298N a pin digital43. Controla el motor 2
27. #define IN4 42                       // IN4 de L298N a pin digital 42. Controla el motor
28.
29. //---De Sensores---
30.
31. #define SEALEVELPRESSURE_HPA (1013.25)    //Nivel del mar base que se utilizara para medir la
    Altitud desde esta referencia
32. #define GASSENSORPIN2 A0                  //Pin analógico utilizado para medir el gas del
    sensor MQ-2
33. #define GASSENSORPIN7 A1                  //Pin analógico utilizado para medir el gas del
    sensor MQ-7
34. #define LIGHTSENSORPIN A7                 //Pin analógico utilizado para medir la luminosidad
    del sensor TEMT6000
35.
36. //--De Bluetooth---
37.
38. #define BTRX  11                          //Pin de Arduino en el que se hará la recepción de
    datos del HC-05
39. #define BTTX  10                          //Pin de Arduino en el que se hará el envió de datos
    del HC-05
40.
41.
42. //------VARIABLES GLOBALES-------
43.
44.
45. Adafruit_BME280 bme;                      //Variable creada de una clase que contiene todas
    las funciones de un sensor BME280
46. SoftwareSerialmyBT(BTRX,BTTX);           //Variable creada de una clase que contiene todas las
    fucniones de un Serial, en este caso utilizado para bluetooth
47. char BTvalue;                             //Variable que guardara los datos que vienen de los
    sensores, que vienen desde la APP
48.
49.
50.
51. //----DECLARACION DE FUNCIONES----
52.
53.
54.   void Movement();                        //Mueve el vehículo en las 4 direcciones, comandada
    desde la APP a traves de Bluetooth
55.   void EnviromentalStation();             //Envía y recibe datos de los sensores, de los
    parámetros ambientales: Temperatura, Humedad, Presión, Altitud, Humo y Monóxido de Carbono
56.
```

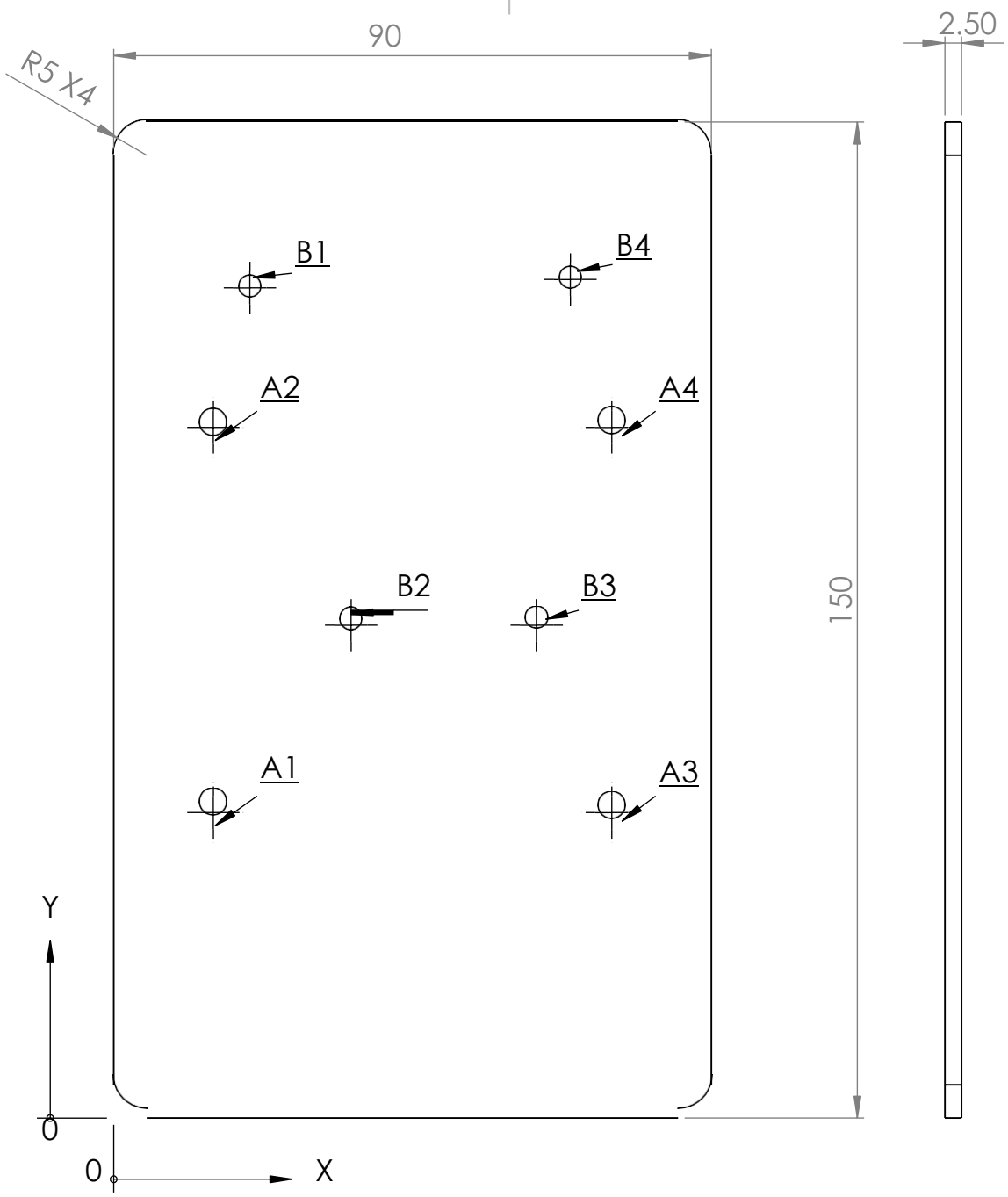
```
57. //-----SETUP de INICIALIZACION-----
58.
59.
60. void setup(){
61.   myBT.begin(38400);                      //Inicio el serial del Bluetooth para comenzar la
    comunicacion
62.   Serial.begin(9600);                     //Inicio el serial de Arduino IDE
63.
64.   pinMode(IN1, OUTPUT);                   //Defino como pines de salida, las 4 entradas que
    controlaran al Puente H y asi, a los 2 motores
65.   pinMode(IN2, OUTPUT);
66.   pinMode(IN3, OUTPUT);
67.   pinMode(IN4, OUTPUT);
68.
69.
70.   pinMode(GASSENSORPIN2,INPUT);           //Defino los pines analógicos de los sensores
    necesarios
71.   pinMode(GASSENSORPIN7,INPUT);
72.   pinMode(LIGHTSENSORPIN, INPUT);
73.
74.
75.   if (!bme.begin(0x76)) {                 //Inicializo la comunicación I2C con el sensor, en la
    dirección en HEX 0X76. Si no encuentra el sensor, hay loop/
76.
77.     Serial.println("Could not find a valid BME280 sensor, check wiring!");
78.     while (1);
79.   }
80.
81. }
82.
83.
84. //-----FUNCION PRINCIPAL EN LOOP----
85.
86.
87. void loop()
88. {
89.
90.       Movement();                         //Repito las funciones constantemente: una controla el
    movimiento
91.       EnviromentalStation();              //Y la otra controla los sensores
92. }
93.
94.
95.
96.
97. //           FUNCIONES
98.
99.
100.
101.       /* Funcion "Movement()"
102.       Entradas: Void
103.       Salidas:  Void
104.
105.       Descripción: Esta función es la que se encarga de recibir del módulo Bluetooth, los
    movimientos indicados para que así, de las instrucciones
106.       correspondientes a los motores y estos puedan moverse en la dirección deseada. Esto lo
    realiza mediante la recepción de "chars" diferentes,
107.       cada vez que se manda la señal desde la APP para que se mueva en cierta dirección el
    robot
108.
109.       */
110.
111.
112.
113.       void Movement(){
114.
115.
116.           if( myBT.available())
              //Si hay una conexionbluetooth, entra en el if
117.                         {
118.
119.
```

```
120.                                        BTvalue=myBT.read();
            //Leo el valor enviado por la app, y decodifico que acción quiere que realice
121.
122.                                        if(BTvalue=='A')
            //Movimiento para ADELANTE
123.                                          {
124.                                          digitalWrite(IN1, HIGH);
125.                                          digitalWrite(IN2, LOW);
126.                                          digitalWrite(IN3, HIGH);
127.                                          digitalWrite(IN4, LOW);
128.
129.                                          }
130.                                        if(BTvalue=='B')
            //Movimiento hacia la DERECHA
131.                                        {
132.                                         digitalWrite(IN2, HIGH);
133.                                         digitalWrite(IN1, LOW);
134.                                         digitalWrite(IN4, HIGH);
135.                                          digitalWrite(IN3, LOW);
136.
137.
138.                                        }
139.                                         if(BTvalue=='C')
            //Movimiento hacia la IZQUIERDA
140.                                          {
141.                                          digitalWrite(IN1, LOW); // IN1 a 0
142.                                          digitalWrite(IN2, HIGH);
143.                                          digitalWrite(IN3, HIGH);
144.                                          digitalWrite(IN4, LOW);
145.
146.                                          }
147.                                         if(BTvalue=='D')
148.                                          {
149.                                          digitalWrite(IN1, HIGH); // IN3 a
    0                           //Movimiento hacia ATRAS
150.                                          digitalWrite(IN2, LOW);
151.                                          digitalWrite(IN3, LOW);
152.                                          digitalWrite(IN4, HIGH);
153.
154.                                          }
155.
156.                                         if(BTvalue=='0')
           //La app envia un 0, cuando se suelta el boton de presionar cualquier tecla de
    movimiento
157.                                         {
158.                                          digitalWrite(IN1, LOW); // IN3 a 0
159.                                          digitalWrite(IN2, LOW);
160.                                          digitalWrite(IN3, LOW);
161.                                          digitalWrite(IN4, LOW);
162.                                         }
163.                               }
164.
165.         }
166.
167.
168.       /* Funcion "EnviromentalStation()"
169.       Entradas: Void
170.       Salidas: Void
171.
172.       Descripción: Esta función es la que se encarga de enviar, a través del modulo
    Bluetooth, los valores de los sensores conectados a Arduino,
173.       para que esta información pueda se accedida en la app. También realiza los cálculos
    necesarios para poder interpretar esa información de la
174.       forma correspondiente y en la unidad correspondiente.
175.
176.       */
177.
178.
179.        void EnviromentalStation()
180.        {
181.
```

```
182.                     float lreading = analogRead(LIGHTSENSORPIN);
          //Leo el Sensor de Luminosidad TEMT6000
183.                     float l_ratio = lreading / 1023.0;

184.                     l ratio = pow(l ratio, 2.0);
          //Realizo los cálculos correspondientes para leer el valor analógico
185.
186.
187.                     int   adc_MQ2 = analogRead(GASSENSORPIN2);
          //Leemos la salida analógica del MQ-2
188.                     float mq2voltage = adc MQ2 * (5.0 / 1023.0);
          //Convertimos la lectura en un valor de mq2voltage
189.                     float Rs2=1000*((5-
    mq2voltage)/mq2voltage);                                    //Calculamos Rs con un RL de 1k,
    basado en el voltaje
190.                     double COgml=95.292*pow(Rs2/5463,-
    1.556)/1000;                                  //Calculamos la concentración de Monóxido de
    Carbono con la ecuación obtenida.
191.
192.                     int   adc MQ7 = analogRead(GASSENSORPIN2);
          //Leemos la salida analógica del MQ-7
193.                     float mq7voltage = adc MQ7 * (5.0 / 1023.0);
          //Convertimos la lectura en un valor de mq2voltage
194.                     float Rs7=1000*((5-
    mq7voltage)/mq7voltage);                                    //Calculamos Rs con un RL de 1k
195.                     double GCgml=644.83*pow(Rs7/5463,-
    2.014)/1000;                                  //Calculamos la concentración de Gases
    Combustibles con la ecuación obtenida.
196.
197.                     myBT.print(bme.readTemperature());
          //Leo y enviopot Bluetooth a la App, las lecturas de los diferentes sensores.
198.                     myBT.print("
    C");                                                                //Printeo también las
    unidades de cada una
199.                     myBT.print("|");
          //NOTA: Esta barra sirve como separación, para que, en la APP, esta sepa cuando tiene
    que
200.                     myBT.print(bme.readHumidity());
          //separar las variables, y asi poder guardarlas y mostralas por separado.
201.                     myBT.print(" %");
202.                     myBT.print("|");
203.                     myBT.print(bme.readPressure());
204.                     myBT.print(" hpa");
205.                     myBT.print("|");
206.                     myBT.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
207.                     myBT.print(" m");
208.                     myBT.print("|");
209.                     myBT.print((int)l_ratio);
210.                     myBT.print(" %");
211.                     myBT.print("|");
212.                     myBT.print((int)COgml);
213.                     myBT.print(" g/ml");
214.                     myBT.print("|");
215.                     myBT.print((int)GCgml);
216.                     myBT.print(" g/ml");
217.                     myBT.print("|");
218.
219.                 return 0;
220.         }
```

# ANEXO B: Dibujos de Piezas




| RÓTULO | UBIC X | UBIC Y | TAMAÑO |
|---|---|---|---|
| A1 | 15 | 46 | ∅4 THRU |
| A2 | 15 | 104 | ∅4 THRU |
| A3 | 75 | 46 | ∅4 THRU |
| A4 | 75 | 104 | ∅4 THRU |
| B1 | 20.51 | 125.02 | ∅3.27 THRU |
| B2 | 35.75 | 74.22 | ∅3.27 THRU |
| B3 | 63.69 | 74.22 | ∅3.27 THRU |
| B4 | 68.77 | 126.29 | ∅3.27 THRU |

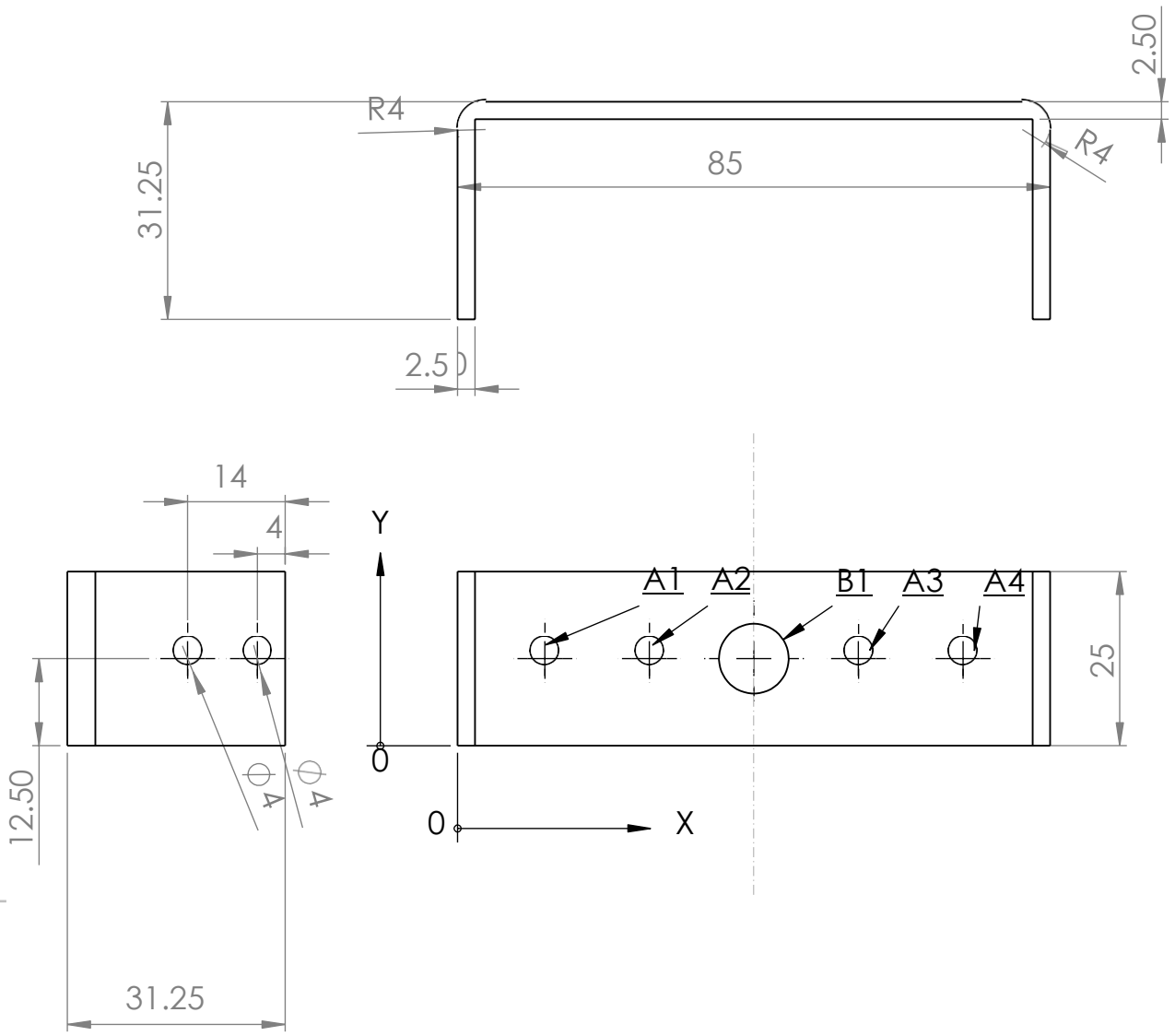
Plano:

# Base 3

A4

ESCALA:1:1

HOJA 1 DE 1



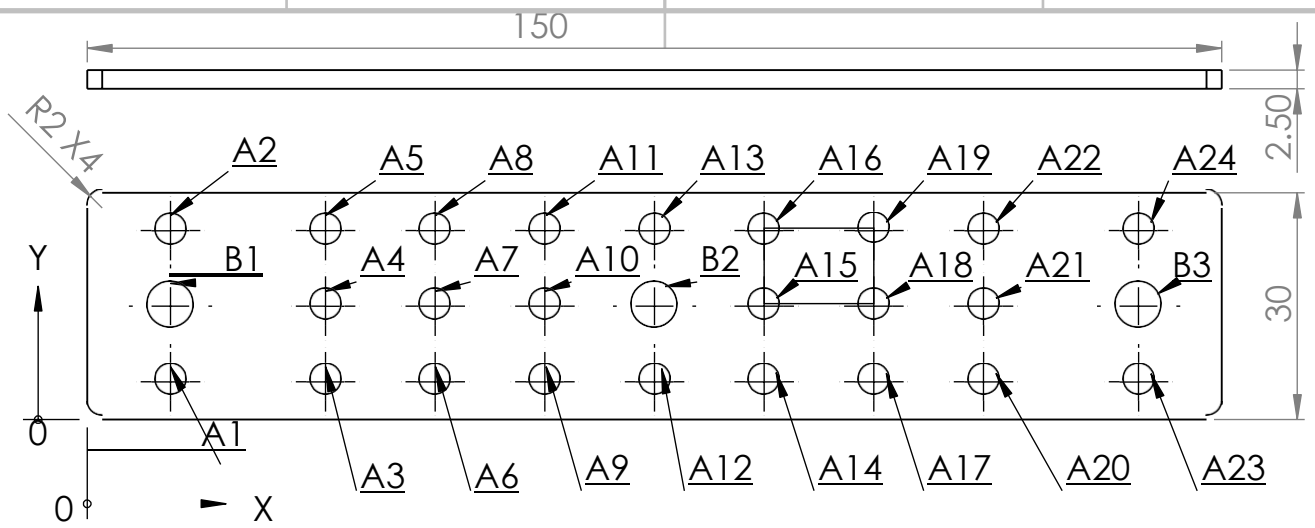

| RÓTULO | UBIC X | UBIC Y | TAMAÑO |
|---|---|---|---|
| A1 | 12.50 | 12.50 | ∅4 THRU |
| A2 | 27.50 | 12.50 | ∅4 THRU |
| A3 | 57.50 | 12.50 | ∅4 THRU |
| A4 | 72.50 | 12.50 | ∅4 THRU |
| B1 | 42.50 | 12.50 | ∅10 THRU |

Plano: Base 2

A4

ESCALA:1:1

HOJA 1 DE 1




| RÓTULO | UBIC X | UBIC Y | TAMAÑO |
|---|---|---|---|
| A1 | 11 | 5 | ∅4 THRU |
| A2 | 11 | 25 | ∅4 THRU |
| A3 | 31.50 | 5 | ∅4 THRU |
| A4 | 31.50 | 15 | ∅4 THRU |
| A5 | 31.50 | 25 | ∅4 THRU |
| A6 | 46 | 5 | ∅4 THRU |
| A7 | 46 | 15 | ∅4 THRU |
| A8 | 46 | 25 | ∅4 THRU |
| A9 | 60.50 | 5 | ∅4 THRU |
| A10 | 60.50 | 15 | ∅4 THRU |
| A11 | 60.50 | 25 | ∅4 THRU |
| A12 | 75 | 5 | ∅4 THRU |
| A13 | 75 | 25 | ∅4 THRU |
| A14 | 89.50 | 5 | ∅4 THRU |
| A15 | 89.50 | 15 | ∅4 THRU |
| A16 | 89.50 | 25 | ∅4 THRU |
| A17 | 104 | 5 | ∅4 THRU |
| A18 | 104 | 15 | ∅4 THRU |
| A19 | 104 | 25 | ∅4 THRU |
| A20 | 118.50 | 5 | ∅4 THRU |
| A21 | 118.50 | 15 | ∅4 THRU |
| A22 | 118.50 | 25 | ∅4 THRU |
| A23 | 139 | 5 | ∅4 THRU |
| A24 | 139 | 25 | ∅4 THRU |
| B1 | 11 | 15 | ∅6 THRU |
| B2 | 75 | 15 | 6 THRU |
| B3 | 139 | 15 | ∅6 THRU |

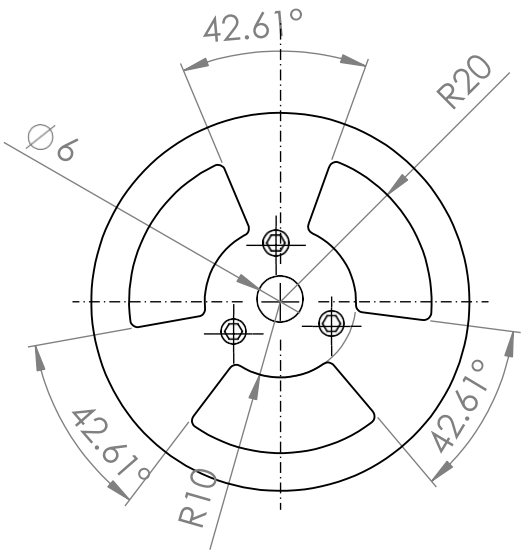
Plano:

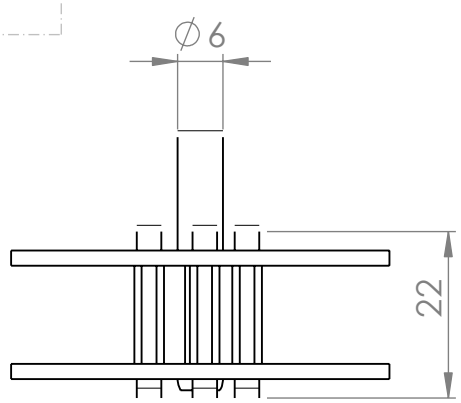
# Base 1

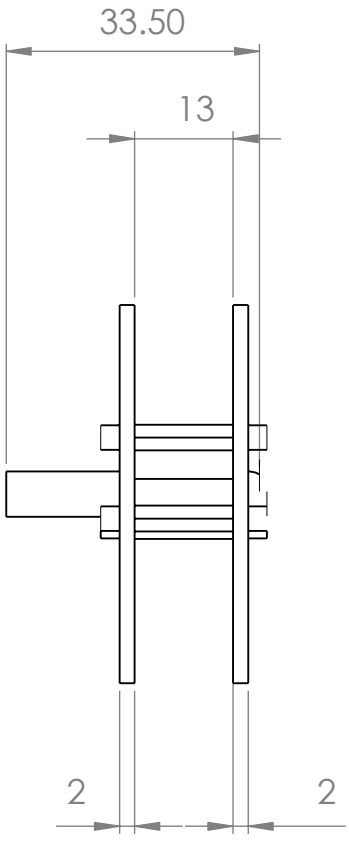
A4

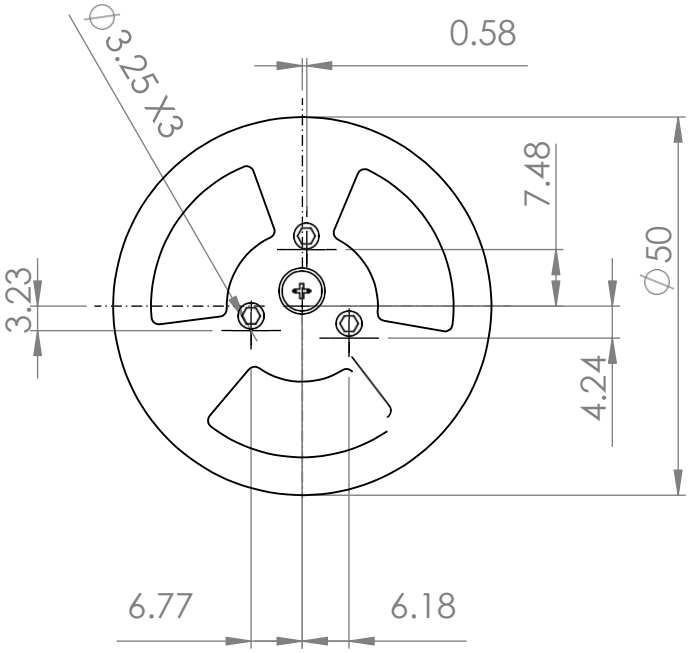
ESCALA:1:1

HOJA 1 DE 1




Plano:

# Rueda

A4

ESCALA:1:1

HOJA 1 DE 1