UNIVERSIDAD NACIONAL DE LA MATANZA

Departamento de Ingeniería e Investigaciones Tecnológicas

Cátedras de:

Sistemas Operativos (Plan 1997- Código 625)

Sistemas Operativos Avanzados (Plan 2009 - Código 1123)

Equipo de docentes: Graciela De Luca, Waldo A. Valiente, Sebastián Barillaro, Mariano Volker, Esteban Andrés Carnuccio y Gerardo García.

Comisión:1900 (lunes noche)

TRABAJO PRÁCTICO:
IoT (SE y Android)

Integrantes grupo Bicicleta Inteligente:

DNI	Apellidos	Nombres
34705098	Vergara	Belen
24715197	Carabajal	Rubén Javier
33901015	Romano	Dario
36159693	Verdicchio	Nicolas
35169931	Jerez	Alex Lionel

Bicicleta Inteligente

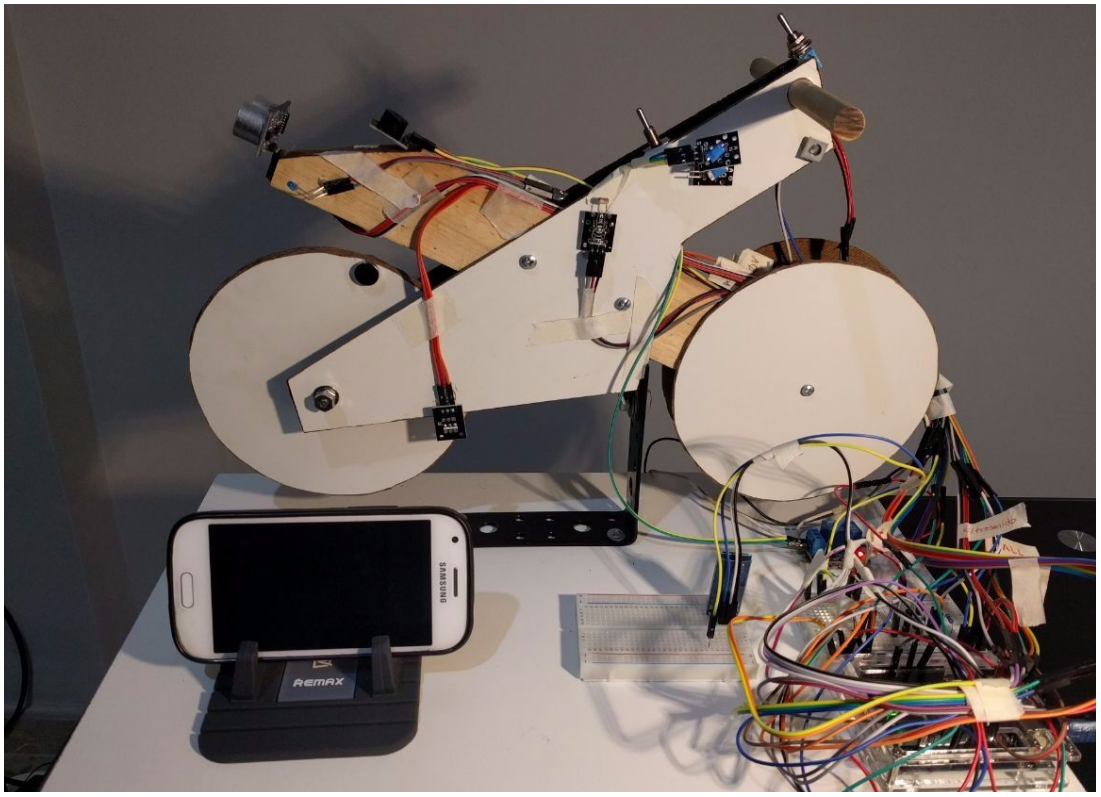


TABLA DE CONTENIDOS

Descripción general

Este proyecto se basa en la integración de varios sensores y actuadores trabajando en conjunto para la mejora en la experiencia de uso en una bicicleta.

Se proponen diferentes mejoras, pensando en la practicidad y en el atractivo de las mismas con cara al usuario. Ofreciendo una mayor integración con los sistemas informáticos actuales. Implementando un sistema práctico y sencillo.

La integración consta de un sistema embebido en una bicicleta, el cual puede trabajar en forma autónoma, con una aplicación (App en Android) que ofrece una interfaz amigable para el usuario

A continuación se realiza una descripción funcional del sistema:

- Información en tiempo real durante el viaje - Visible en App
  - Tiempo transcurrido
  - Velocidad de viaje
  - Distancia recorrida
  - Mi ubicación en el mapa

- Alertas en tiempo real - Visible en App
  - Durante el viaje: objeto cercano detrás de la bici
  - Bicicleta estacionada con la alarma activada: alarma de detección de movimiento (alerta ante posible robo)

- Mecanismos automáticos
  - Si hay poca luz, enciende la iluminación del chasis (para que sea fácil ver la bicicleta)
  - Deshabilitación de iluminación cuando se detecta fin de viaje o activación de alarma.

- Mecanismos Manuales
  - Luces de giro
  - Opcional manual: Deshabilitación / Apagado de iluminación de chasis
  - Opcional manual: Habilitación / Deshabilitación de Alarma.

- Integración con Android - Desde App.
  - Display de información en tiempo real
  - Display de configuración que me permite apagar/encender las funcionalidades de “objeto cercano” y “luz automática del chasis”.
  - Control de alarma

Ventajas de uso

- Ventajas
    - El ciclista no necesita mirar hacia atrás para saber si tiene algún vehículo cerca sobre su carril.
    - El sistema previene accidentes detectando baja luminosidad y encendiendo automáticamente las luces del chasis.
    - Al tener un modo alarma, el ciclista puede sentirse más seguro al dejar su bicicleta estacionada.
    - El sistema mejora comunicación con el medio al indicar el sentido de circulación.
    - El sistema tiene la capacidad de detectar una caída y proveer herramientas para solicitar servicios de emergencia fácilmente.

Especificaciones técnicas - ARDUINO

Diagrama esquemático

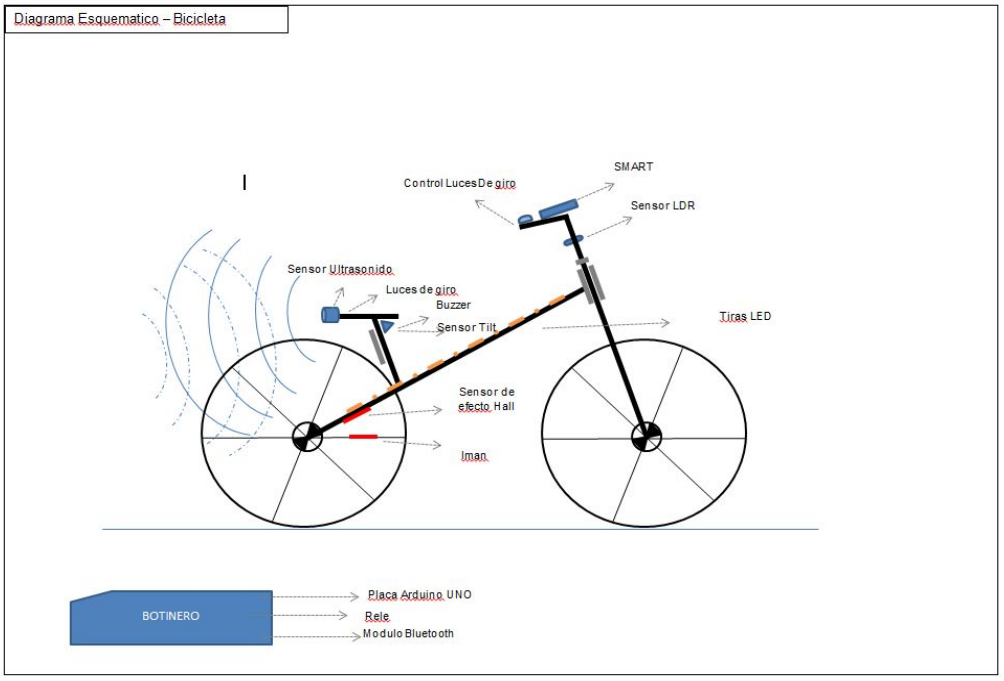


Diagrama de señales

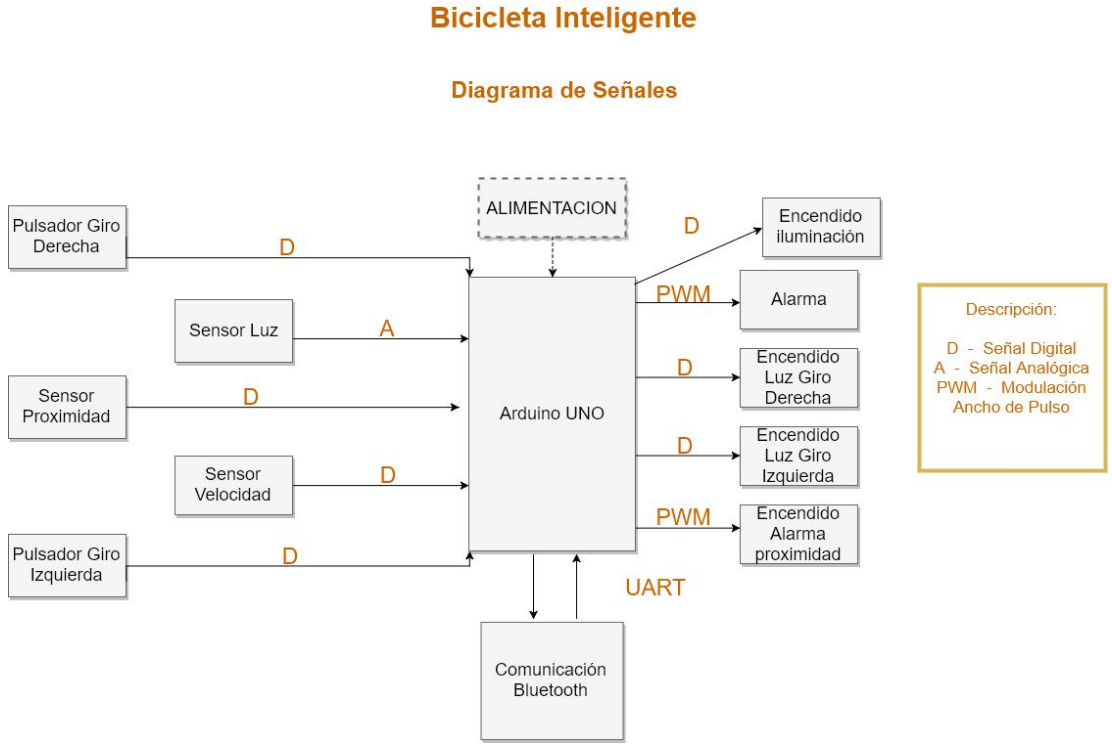


Diagrama de componentes

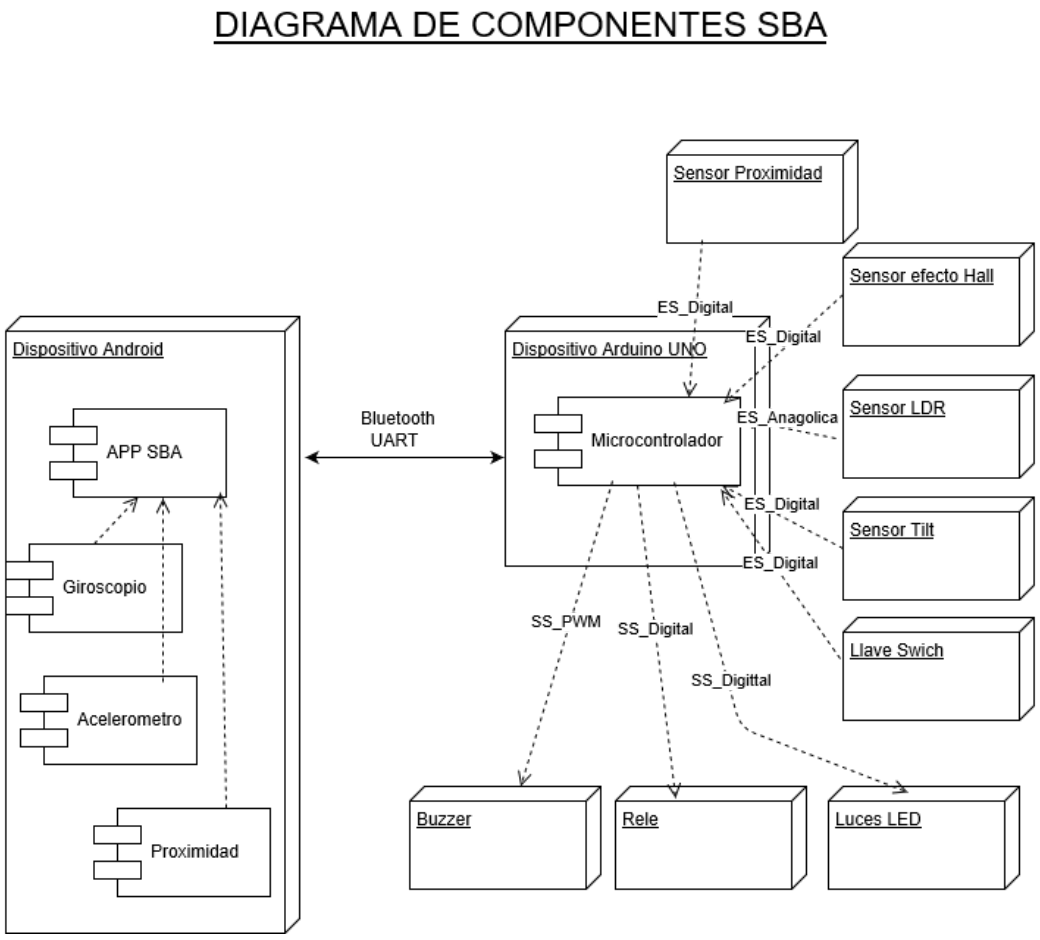
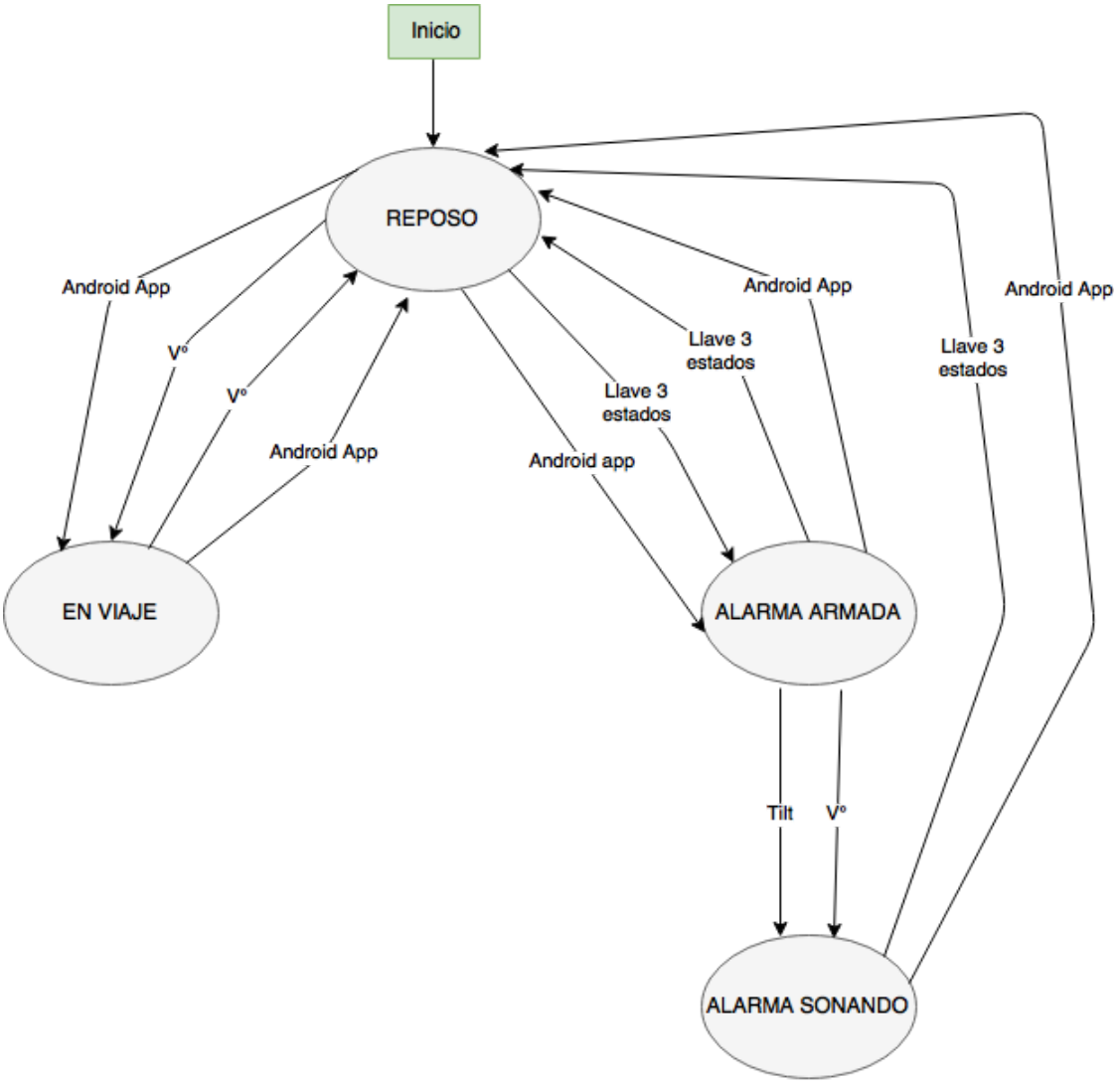


Diagrama de estados

Las funcionalidades del SE dependen del estado en el que se encuentre el mismo (máquina de estados).

El siguiente diagrama ilustra los 4 estados posibles del sistema y las transiciones entre los mismos.

Cada transición (flecha) menciona al sensor (o aplicación) que dió origen al cambio.



Estado reposo

Al encender la bicicleta, se inicia en estado de reposo/stand by. La responsabilidad de éste es esperar acciones o eventos que generen un cambio de estado.

¿Qué podría suceder en el sistema para irme de REPOSO?

- El usuario comienza un nuevo viaje. Formas de detectar este evento:
    - Validando la velocidad de la bicicleta: si se superan los 5 km/h (velocidad aproximada que utiliza un humano para caminar), consideramos que comenzó el viaje y el sistema pasa a EN VIAJE.
    - Recibiendo un mensaje BT: si el usuario además está usando nuestra aplicación y presiona en “Comenzar viaje”, Android le avisa al SE de este evento y pasa al sistema a EN VIAJE.
- El usuario activa la alarma. Formas de detectar este evento:
    - Validando las combinaciones ingresadas en la llave de 3 estados. Si el usuario ejecuta derecha-derecha-derecha, el sistema pasa a ALARMA ARMADA.
    - Recibiendo un mensaje BT: si el usuario además está usando nuestra aplicación y presiona en el icono de la alarma, Android le avisa al SE de este evento y pasa al sistema a ALARMA ARMADA.

Estado en viaje

El usuario se encuentra camino a su destino. Este estado tiene asociadas las siguientes funcionalidades:

- Objeto cercano: el sistema utiliza un sensor de proximidad para poder detectar objetos cercanos. Si el objeto (que podría ser un auto) se encuentra a menos de 50cm, el sistema lanza un pitido como para dar aviso al usuario. Si el usuario además está usando nuestra aplicación, el SE envía un mensaje de aviso al dispositivo Android indicando que hay un objeto cercano.
- Encendido de luces del chasis. Formas de ejecutar esta acción:
    - Automático: el sensor de luz detecta baja luminosidad y el sistema decide encenderlas.
    - Manual: este es un caso particular. Puede suceder que por calor recibido por el sensor, éste no funcione como esperamos. Entonces, si es de noche y las luces todavía no se encendieron, el usuario tiene una llave para encenderlas manualmente.
- Luces de dirección: utilizando la llave de 3 estados, el usuario puede avisar a su entorno hacia dónde está por doblar y así, el sistema, encenderá el led correspondiente (parpadeo cada 500 ms).
- Display de información en tiempo real: si el usuario además está usando nuestra aplicación y se encuentra en la pantalla de tiempo real (similar al tablero de un auto), podrá ver la siguiente información del viaje
    - Tiempo transcurrido
    - Distancia recorrida
    - Velocidad de la bicicleta
    - Mi ubicación en el mapa
    - Alerta de llamado de emergencia en el caso que el usuario se caiga de la bicicleta

¿Qué podría suceder en el sistema para irme de EN VIAJE?

- El usuario finaliza su viaje. Formas de detectar este evento:
    - Validando la velocidad de la bicicleta: si la bicicleta está detenida (velocidad cero) por al menos 1 minuto, consideramos que el viaje terminó y el sistema pasa a REPOSO.
    - Recibiendo un mensaje BT: si el usuario además está usando nuestra aplicación y presiona el botón de finalizar viaje, Android le avisa al SE de este evento y pasa al sistema a REPOSO.

Estado alarma armada

La alarma se encuentra activa.

¿Qué podría suceder en el sistema para irme de ALARMA ARMADA?

- El usuario desactiva la alarma. Formas de ejecutar esta acción:
  - Validando las combinaciones ingresadas en la llave de 3 estados. Si el usuario ejecuta izquierda-izquierda, la alarma se desactiva y el sistema pasa a REPOSO.
  - Recibiendo un mensaje BT: si el usuario además está usando nuestra aplicación y presiona en el icono de la alarma, Android le avisa al SE de este evento y pasa al sistema a REPOSO.
- La alarma empieza a sonar. Formas de generar este evento:
  - Validando al sensor tilt: si la bicicleta se encuentra inclinada por un determinado tiempo (dependiente de un contador del sistema que considera el efecto rebote), el sistema hace sonar la alarma (alterando la frecuencia de un buzzer pasivo).
  - Validando la velocidad de la bicicleta: si se superan los 3 km/h, consideramos que alguien se está llevando la bicicleta y el sistema pasa a ALARMA SONANDO.

Estado alarma sonando

La alarma está sonando, a la espera de ser desactivada. Además, este estado enciende las luces del chasis como para llamar más la atención.

¿Qué podría suceder en el sistema para irme de ALARMA SONANDO?

- El usuario apaga la alarma. Formas de realizar esta acción:
  - Validando las combinaciones ingresadas en la llave de 3 estados. Si el usuario ejecuta izquierda-izquierda, el sistema pasa a REPOSO.
  - Recibiendo un mensaje BT: si el usuario además está usando nuestra aplicación y presiona en el icono de la alarma, Android le avisa al SE de este evento y pasa al sistema a REPOSO.

Lista de materiales

- ✓ Placa Arduino UNO Rev 3
- ✓ Sensor ultrasonidos HC-SR04
- ✓ Modulo Bluetooth HC-05
- ✓ Sensor efecto Hall (KY-003 - 3144 402)
- ✓ Tilt switch (KY-020)
- ✓ Módulo fotoresistor (KY-018)
- ✓ Llave de 3 estados
- ✓ Llave de 2 estados
- ✓ 2 indicadores LED
- ✓ Buzzer pasivo (KY-006)
- ✓ Relé 5 VDC (KY-019)
- ✓ Fuente 12V
- ✓ Tira de LED’s de 12VDC 2A
- ✓ Imán de neodimio 5x1 mm
- ✓ Dispositivo móvil con Android (versión mínima 4.4 Kit-Kat)
- ✓ Protoboard de 830 puntos
- ✓ Protoboard de 400 puntos

Entradas digitales

- ✓ Llave de 3 estados
- ✓ Sensor efecto Hall
- ✓ Tilt switch
- ✓ Sensor ultrasonido

Salidas digitales

- ✓ Control de Relé
- ✓ Control de LED's de giro

Entradas Analogicas

- ✓ Fotoresistor

Salidas Analogicas

- ✓ Alarma (Buzzer pasivo)

Sensores y actuadores - Detalle técnico

Sensor ultrasonidos HC-SR04.

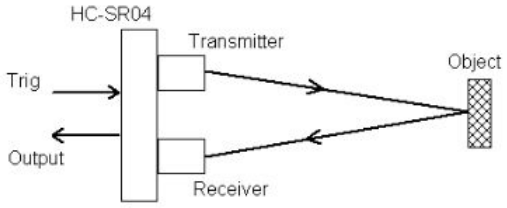
El sensor de ultrasonidos se enmarca dentro de los sensores para medir distancias o superar obstáculos, entre otras posibles funciones.
En este caso vamos a utilizarlo para la medición de distancias.

Principio de funcionamiento

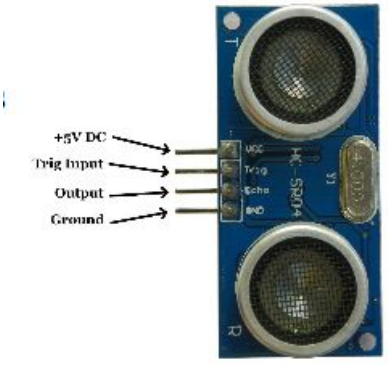
Esto lo consigue enviando un ultrasonido (inaudible para el oído humano por su alta frecuencia) a través de uno de la pareja de cilindros que componen el sensor (un transductor) y espera a que dicho sonido rebote sobre un objeto y vuelva, retorno captado por el otro cilindro.

El HC-SR04 *consta de transmisor ultrasónico, receptor y circuitos de control.*

Cuando se dispara (detecta un nivel alto sobre el pin Trig), envía una serie de pulsos de ultrasonidos de 40KHz y recibe eco (fenómeno acústico producido cuando una onda se refleja y regresa hacia su emisor) de un objeto. La distancia entre la unidad y el objeto se calcula midiendo el tiempo de desplazamiento del sonido y poniéndolo como el ancho de un pulso TTL.



Para medir la distancia, el microcontrolador necesita generar una señal de disparo y conducirla al pin de entrada del disparo del sensor de ultrasonido. El nivel de la señal de disparo debe cumplir los requisitos de nivel del TTL (es decir, **nivel alto > 2,4 V, nivel bajo < 0,8 V**) y su ancho debe ser superior a 10 us. Al mismo tiempo, es necesario supervisar el pin de salida midiendo el ancho de pulso de la señal de salida. La distancia detectada se puede calcular mediante una fórmula.



Aprovechando que la velocidad de dicho ultrasonido en el aire es de valor **340 m/s (velocidad del sonido) = 0,034 cm/microseg** (ya que trabajaremos con centímetros y microsegundos). Para calcular la distancia, recordaremos que **v = d / t** (definición de velocidad: distancia recorrida en un determinado tiempo).

$$Distancia = \frac{(Trigger - Echo) * (340m/s)}{2}$$

De la fórmula anterior despejamos d, obteniendo d = v · t, siendo v la constante anteriormente citada y t el valor devuelto por el sensor a la placa Arduino.
También habrá que dividir el resultado entre 2 dado que el tiempo recibido es el tiempo de ida y vuelta.

Especificaciones Técnicas ultrasonido

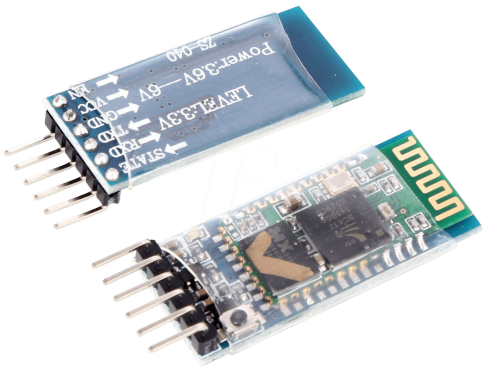
- Dimensiones del circuito: 43 x 20 x 17 mm
- Tensión de alimentación: 5 Vcc
- Frecuencia de trabajo: 40 KHz
- Rango máximo: 4.5 m
- Rango mínimo: 1.7 cm
- Duración mínima del pulso de disparo (nivel TTL): 10 µS.
- Duración del pulso eco de salida (nivel TTL): 100-25000 µS.
- Tiempo mínimo de espera entre una medida y el inicio de otra 20 mS.

Problemas con las Mediciones de Ultrasonido

- La velocidad del sonido se ve afectada por la temperatura (la velocidad de propagación de una onda mecánica decrece a medida que la temperatura aumenta) y por el medio en el que se lo propague.
- El rango de medición debe estar entre 1.7 cm y un máximo de 450 cm.
- La onda de rebote debe estar en rango angular, en donde el ángulo efectivo es de 30° a izquierda, 15° a derecha. Un ángulo mayor a 15°, producto dependiente de la superficie en la que rebote la onda, quedaría fuera del alcance de lectura del receptor, por lo que sería un error en la medición.
- Debe transcurrir al menos 10 µS para que se pueda realizar una lectura de una onda.
- Si existe un gran número de objetos, el sonido rebota en las superficies generando ecos y falsas mediciones.

Modulo Bluetooth HC-05

Para la comunicación entre el dispositivo móvil y el sistema Arduino trabajamos con el módulo HC-05.



Módulo HC-05

Su principal ventaja es que puede operar una conexión inalámbrica serie de forma transparente para el Arduino.

Para configurar el HC-05 utilizamos los comandos AT para establecer los siguientes parámetros:

- Nombre del dispositivo bluetooth correspondiente al módulo
  - AT+NAME=SBA-2017
- Contraseña para apareo
  - AT+PSWD=1234
- Rol esclavo
  - AT+ROLE=0

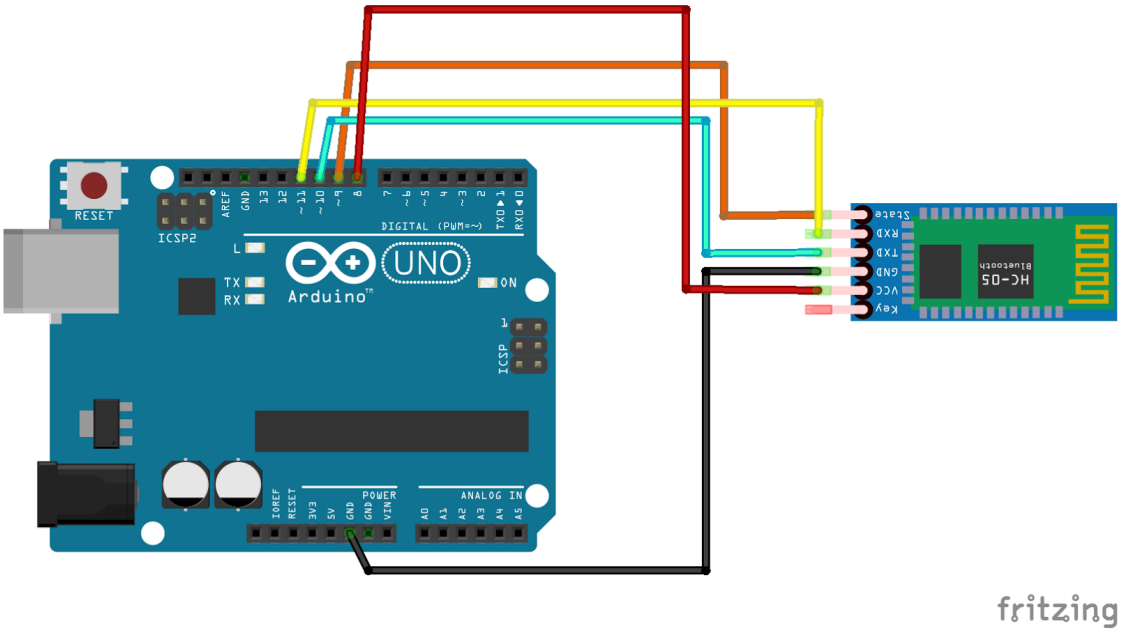
Configuramos el módulo HC-05 como esclavo para que el apareo pueda realizarse desde el dispositivo móvil.

Como limitación, encontramos que la conexión entre el dispositivo móvil y el módulo HC-05 no podría superar los 10 metros para su óptimo funcionamiento. Sin embargo, dada la naturaleza de las funcionalidades del sistema, ese rango nos resulta más que suficiente.

Especificaciones Técnicas módulo HC-05

- Módulo de puerto serie inalámbrico Bluetooth JY-MCU
- El voltaje de la fuente puede estar entre 3.6 a 6V DC. Las E / S son tolerantes a 5V.
- Bluetooth V2.0+EDR (velocidad de datos mejorada) 3Mbps
- Bluetooth SPP (Serial Port Protocol)
- Fácil de conectar este módulo con cualquier dispositivo Bluetooth estándar, sólo la búsqueda y llave "1234" contraseña.
- Velocidad de transmisión : 38400 bps.
- Velocidad en modo de comunicación: 9600 bps.
- El módulo no requiere configuración.
- Protocolo base de Bluetooth 802.15
- Dimensiones: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)

Para nuestro proyecto se planteó la siguiente conexión de prueba:



Protocolo de comunicación - UART

Utilizamos el protocolo de comunicación serie UART (Universal asynchronous receiver-transmitter) dado que al ser de carácter asincrónico, no bloquea el flujo de procesamiento del sketch principal del sistema embebido.

Además, es soportado nativamente por Arduino Uno, ya que el chip Atmega sobre el que está construido contiene internamente una pieza de hardware UART.

Convención de mensajes

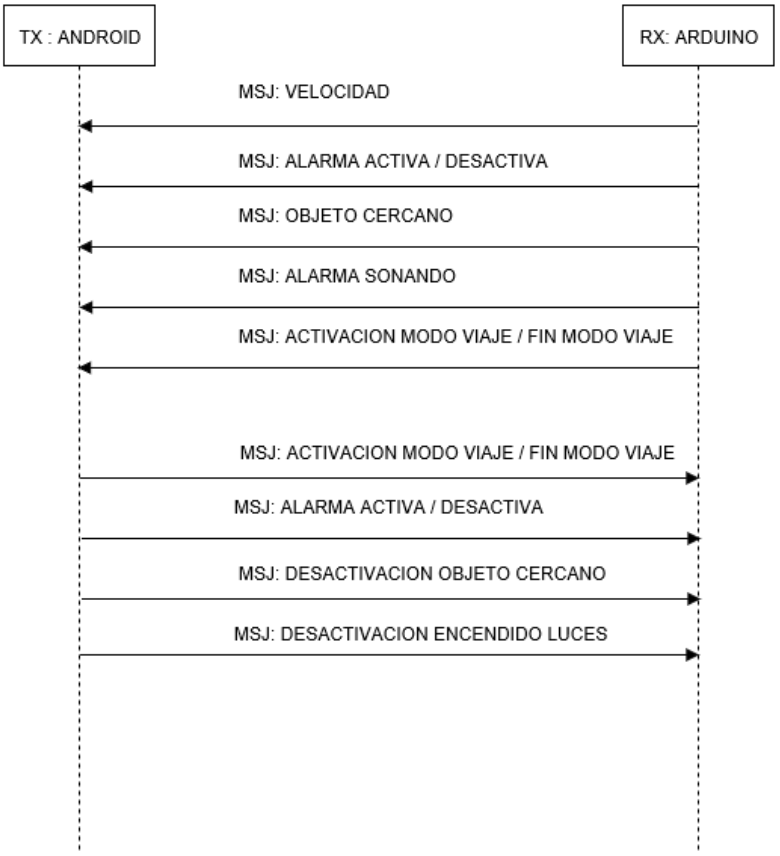
Para el envío de mensajes entre Arduino y Android, utilizamos un entero como identificador. Cada uno de ellos representa una acción a realizar.

Todos los mensajes son finalizados con '\r\n' por convención del HC-05.

Identificador	Descripción
0	Desconocido.
1	Comenzó el viaje
2	Finalizó el viaje
3	Activar alarma
4	Desactivar alarma
5	La alarma está sonando
6	Apagar alarma sonando
7	Transmite velocidad
8	Objeto cercano
9	Habilitar funcionalidad objeto cercano
10	Deshabilitar funcionalidad objeto cercano
11	Habilitar funcionalidad luz del chasis automática
12	Deshabilitar funcionalidad luz del chasis automática
700	La bicicleta está detenida (velocidad cero)
[703-750]	Este rango de enteros está reservado para transmitir la velocidad actual de la bicicleta (en modo viaje). ¿Cómo se interpreta? Cuando la bicicleta está andando, nuestra biblioteca LibVelocidad siempre nos va a devolver un valor entre 3 km/h y 50 km/h. Para

	poder transmitir ese valor, elegimos un valor entero base (700 en este caso). Antes de transmitir, le sumamos 700 a la velocidad. Entonces, el receptor del mensaje (Android), lo único que tiene que hacer es quedarse con las unidades y decenas del número recibido (es tan simple como restarle el número base al número recibido).

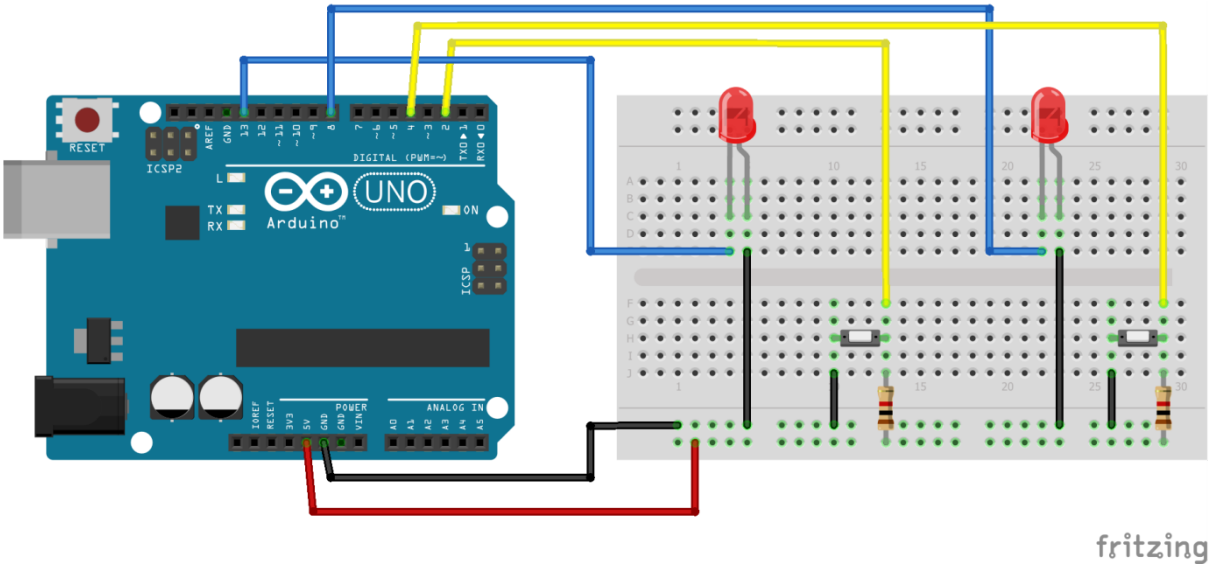
MENSAJES ANDROID - ARDUINO



Módulo indicador sentido de dirección

Este módulo se encarga de encender y apagar las luces de giro de la bicicleta durante el modo viaje. El ciclista indica la dirección de giro presionando el pulsador correspondiente y el sistema comienza a realizar un ciclo de intermitencia sobre el LED indicado.

Para nuestro proyecto se planteó la siguiente conexión de prueba:



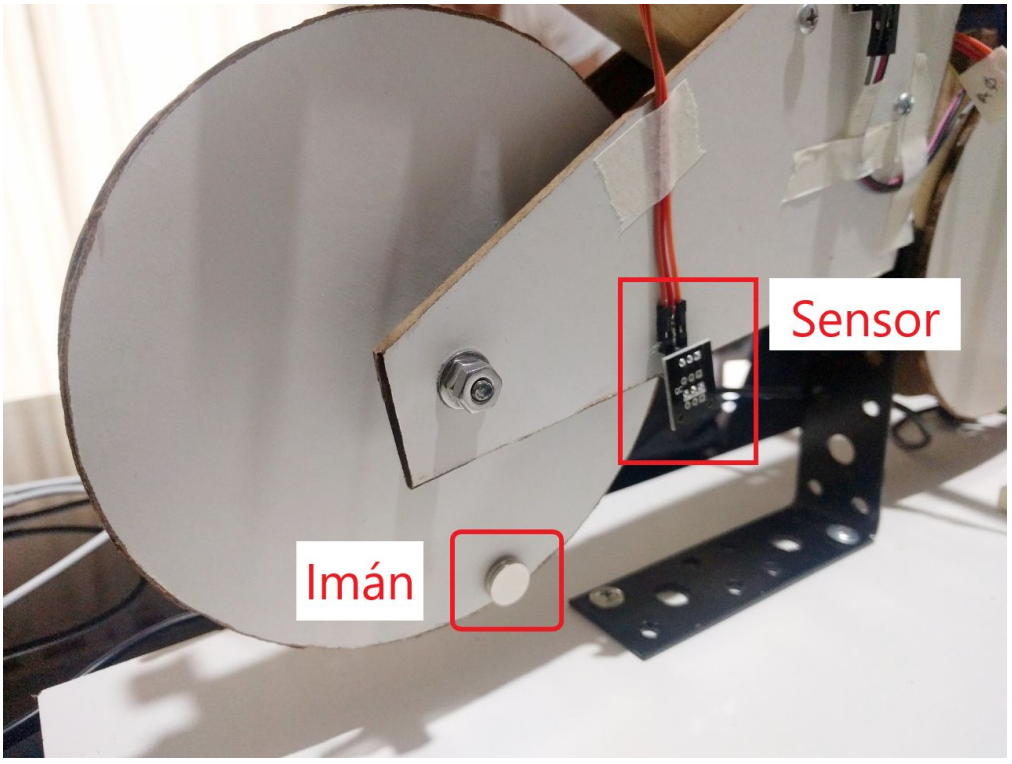
El circuito final implementado en el prototipo utilizar una llave de 3 estados en lugar de 2 pulsadores.

Módulo de medición de velocidad

El módulo de medición de velocidad es utilizado durante el modo viaje para enviar la velocidad calculada de la bicicleta a la aplicación móvil. Durante el modo de alarma activada se lo utiliza para detectar que la bici no esté en movimiento. Por último, en el estado reposo, se utiliza para detectar que comenzó el viaje.

Construcción

Montado sobre la rueda trasera colocamos 2 imanes de neodimio apilados (uno solo también es funcional). Sobre el chasis, a 90% el sensor de efecto Hall.



Componentes

- Sensor efecto Hall (KY-003 - 3144 402)
- Imán neodimio 5x1 mm

Funcionamiento

El sistema cuenta la cantidad de vueltas por minuto que la rueda realiza y (en base a valores preconfigurados) calcula el módulo de la velocidad.

Por defecto, el sistema está configurado para una rueda estándar de 26” (ISO 50-559, 559 mm de diámetro).

La fórmula que el sistema utiliza para realizar el cálculo de la velocidad es la siguiente:

$$v\ [km/h] = \frac{2 \times \pi \times r \times rpm \times 60}{1000}$$

Siendo:

- v = velocidad en km/h
- r = radio de la rueda en metros
- rpm = el número de revoluciones por minuto

Como limitaciones tiene una velocidad de lectura mínima de 3 km/h y una máxima de 50 km/h con un tiempo de respuesta de 2 segundos.

A nivel software, las vueltas se cuentan a partir del registro de un *flanco de bajada* de la señal que entrega el sensor de efecto Hall con un tiempo de medición de 100 ms (esto limita la velocidad máxima a 50 km/h). Se cuentan revoluciones a lo largo de 2 segundos, calcula el promedio de RPMs y reinicia el contador de revoluciones (esto limita la velocidad mínima a 3 km/h). Utilizamos la detección de flancos de bajada para evitar que si la rueda está frenada con el imán sobre el sensor, el sistema no contabilice el nivel alto del sensor como una rotación de la rueda y genere un reporte de velocidad erróneo.

Sensor efecto Hall

Los sensores Hall son dispositivos que permiten realizar mediciones de campo magnético.

Principio de funcionamiento

Para los sensores de efecto Hall analógicos (Linear Hall), si fluye corriente por uno de estos y se aproxima a un campo magnético que fluye en dirección vertical al sensor, entonces el sensor crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente.

Existen sensores de efecto Hall con salida digital, que agregan al funcionamiento de un analógico un mecanismo para entregar una salida discreta de 2 estados. Estos tienen especificaciones de trabajo en las que se establecen umbrales de valores de campo magnético para los que el sensor entregará cada valor de salida.

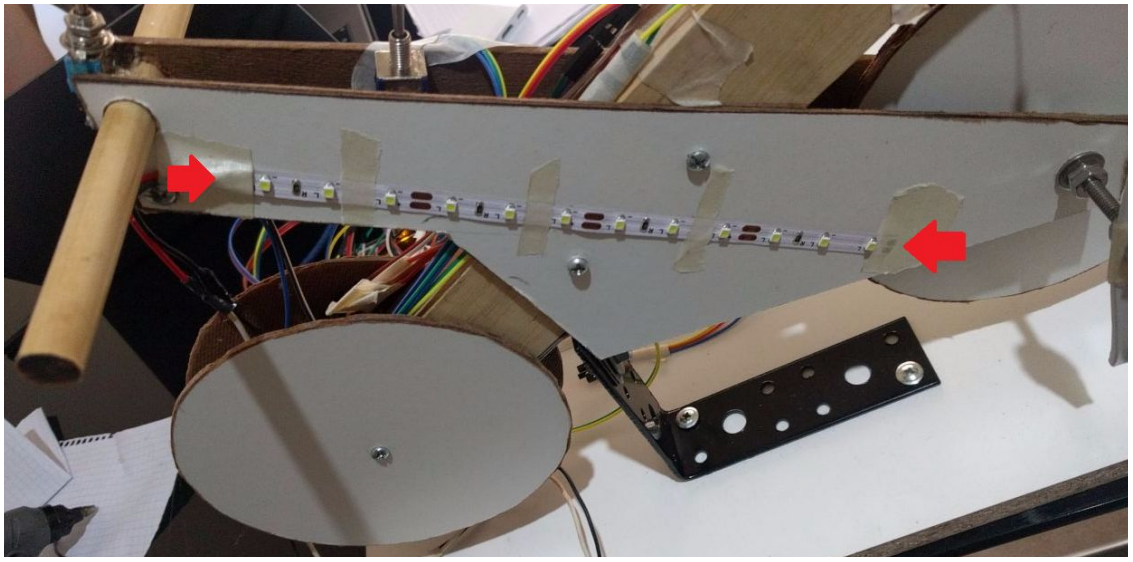
Alguno sensores digitales, incluyen un módulo que proporciona un ciclo de histéresis interna para filtrar y producir una salida más estable.

Módulo de iluminación de chasis

Construcción

Montado sobre un lado del chasis colocamos la iluminación LED y sobre el opuesto el módulo fotoresistor. Con esto evitamos que el actuador (LED's) altere las mediciones del sensor (fotoresistor) de este módulo.

El encendido y apagado de los LED's es controlado desde el Arduino por medio de un relé. Existe una llave conectada en paralelo al mismo para poder encender manualmente la iluminación, dado el caso particular que el módulo no detecte correctamente que el ambiente está oscuro.

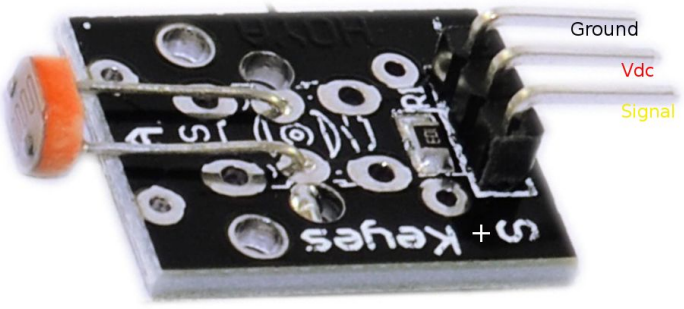


Componentes

- Tira de LED's 12V
- Relé 5 VDC (KY-019)
- Módulo fotoresistor (KY-018)

Fotoresistencia

Utilizamos un módulo fotoresistor (KY-018). Su construcción es la siguiente:



- Ground: 0v

- Vdc: 5V
- Signal: señal de salida

Sobre el conector Signal se obtiene la señal sobre la que se infieren los valores de luminosidad del ambiente.

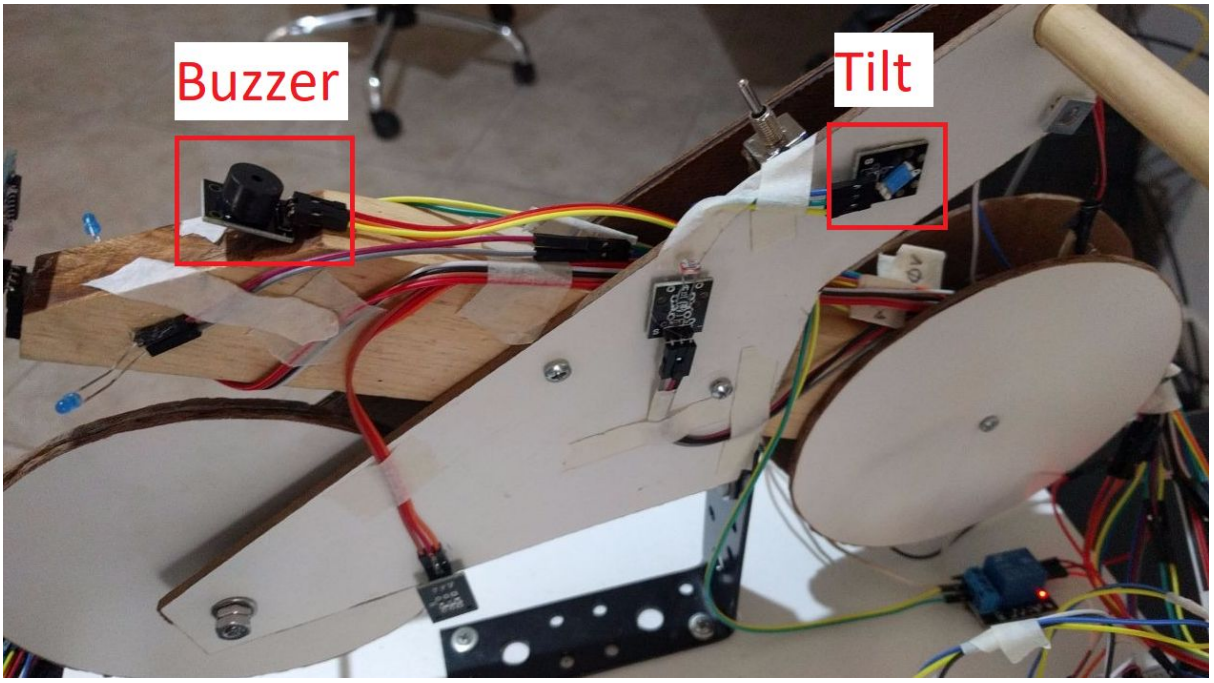
En esencia, basándonos en experimentación, determinamos 2 umbrales de valores de luminosidad para encender y apagar la iluminación LED.

También incorporamos un retardo de acción sobre las lecturas para evitar que sombras o iluminaciones externas pasajeras desencadenaran falsos positivos generando innecesarios cambios de estado sobre el actuador.

Módulo alarma

Construcción

Colocamos el sensor de inclinación (Tilt) sobre una parte elevada del chasis con la finalidad de que se requiera de una inclinación no muy pronunciada para disparar la alarma. El modo alarma se accede mediante el ingreso de una clave con la llave de 3 estados de la luz de giro o por medio de la aplicación móvil.



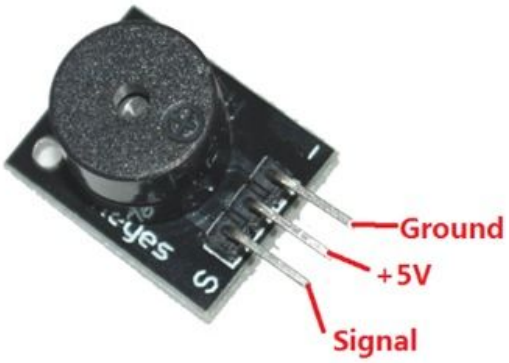
Componentes

- Tilt switch (KY-020)
- Buzzer pasivo (KY-006)

Buzzer pasivo

Utilizamos un buzzer pasivo para poder generar alarmas sonoras con distintos tonos que indican distintas alertas. Esto provee un entendimiento más intuitivo del sistema al ciclista y brinda una experiencia de usuario más satisfactoria.

El buzzer que utilizamos es el siguiente:



Su composición:

- Ground (-): 0v
- +5V: 5v
- Signal (s): señal que provee la frecuencia del tono deseado

Para especificar los tonos que queremos generar, utilizamos la función tone(pin, frequency, duration) indicando el pin de salida del Arduino, la frecuencia de la señal y la duración de la misma. Para detener la salida de la señal utilizamos noTone(pin).

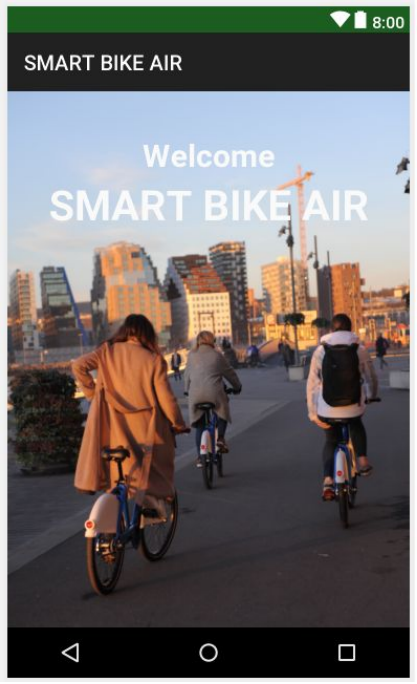
Diseño en android - "smart bike air" sba.

Análisis funcional

1. Verificación conexión BT - Inicio Aplicación

La aplicación mostrará una Activity de inicialización mientras se esté realizando la conexión y sincronización de la aplicación con el dispositivo inteligente en la bicicleta. Una vez cargada por completo accede a la vista principal de inicialización de actividad.

Vista inicial:



2. Home Principal

La Home principal cuenta con 3 accesos rápidos para las funcionalidades de:

- ✓ Start: Inicio de viaje, se accede a la activity RealTime (se detalla funcionalidad en el próximo apartado).
- ✓ Activación / Desactivación Alarma: Se envía mensaje de activación/desactivación de alarma al SE montado en la bicicleta, este cambio de estado se ve reflejado en el texto asociado al button que realiza dicha acción (ACTIVATE = Alarma activa , DEACTIVATE = Alarma Desactivada). En caso que se active la misma del lado del SE el botón cambia su estado a "ACTIVATE"
- ✓ Settings: Se accede a la Activity Preferences para control de Arduino (se detalla en los próximos apartados)

Si la alarma está sonando, la aplicación le avisa de esto al usuario mostrando un diálogo:

3. *Real Time*

3.1. Comenzar viaje

3.1.1. En caso que el usuario no haya ejecutado la Activity por medio de la opción “START”, la misma se iniciará de 2 formas diferentes:

A. Mensaje de activación inició MODO VIAJE: desde el SE se enviará un mensaje indicando el cambio de estado a MODO VIAJE
B. Activación mediante el registro de cambios en el sensor accelerometer: a través del monitoreo del sensor, incluido en el dispositivo android, y cumpliendo los parámetros de modificación en los valores de X, Y y Z se lanzará la Activity Real Time.

3.1.2. Tenemos dos formas de acceder a la ubicación actual del usuario:

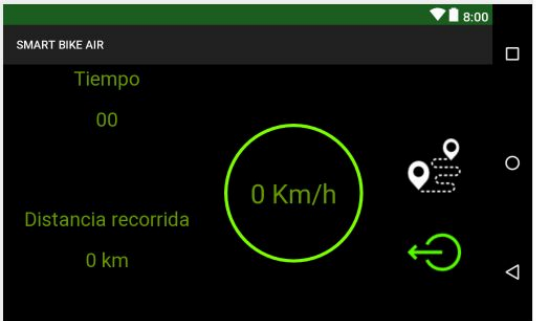
A. Presionando el botón del Mapa (botón blanco).
B. Deslizando la mano por encima de la pantalla (a pocos cm de la pantalla). Esto se obtiene gracias a la utilización del sensor de proximidad. Esta alternativa tiene la ventaja de no necesitar poner foco en la pantalla del celular para buscar al botón.

3.1.3. Cuando nos encontramos en modo viaje si se detecta un movimiento brusco, simulando una caída del ciclista, se despliega un diálogo que nos permitirá realizar una llamada de emergencia. Esta funcionalidad se encontrará disponible si el dispositivo android cuenta con el sensor gyroscope, ya que se utilizan sus variaciones en los registros para poder detectar una variación en el dispositivo que indique una caída.

Se desplegará el siguiente diálogo:

3.1.4. Se visualizará información en tiempo real asociada al viaje en curso:

3.1.4.1. Tiempo
3.1.4.2. Distancia recorrida
3.1.4.3. Velocidad
3.1.4.4. Alerta de objeto cercano (mensaje emergente)
3.1.4.5. Acceso a mi ubicación actual



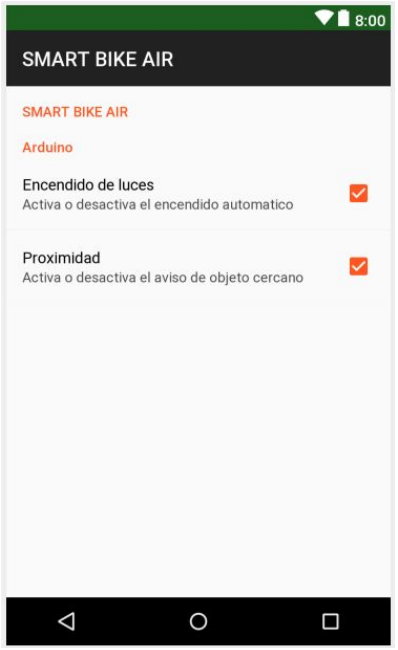
3. Configuración

4.1. Se mostrará un listado con ítems de configuración.

4.2. Configuración de la bicicleta (asociada a los sensores):

4.2.1. Activar/desactivar sistema de luces automático.

4.2.2. Activar/desactivar sistema de detección de objeto cercano.



Sensores utilizados Android

Para utilizar los sensores, utilizamos al Sensor framework. Éste nos brinda las siguientes clases:

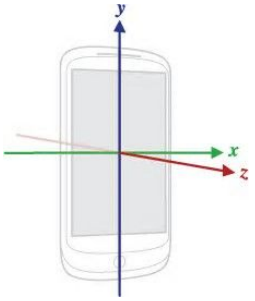
- SensorManager
- Sensor
- SensorEvent
- SensorEventListener

Sensor Acelerómetro

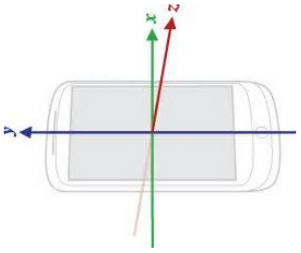
La pantalla Home se encuentra suscrita a los cambios que se detecten en el sensor, permitiendo que, al detectar cierto movimiento, se ejecute la pantalla de RealTime de forma automática.

Teniendo en cuenta que el acelerómetro es un sensor que mide la aceleración relativa a la caída libre como marco de referencia, se establecieron valores mínimos que deben tomar los ejes X y Z para considerar que nos encontramos comenzando un viaje y con el dispositivo se encuentra en la orientación correcta de funcionamiento.

Ejes de uso normal:



Posicionamiento necesario para utilización en el sistema SBA.



Se establecieron los siguientes rangos para considerar que nos encontramos en inicio de viaje y la activación automática de la pantalla Tiempo real.

Eje X:

El eje X se debe encontrar en posición perpendicular a la línea de tierra, de esta forma detectamos que el dispositivo se encuentra en Landscape.
Los valor minimos y maximos, que debe tomar el eje X para considerar el cambio de posición, varían en el rango de X > 9 u X < -9. Estos valores se establecieron tomando como valor base la gravedad, que en la posición Landscape afecta al eje X.

Eje Z:

EL eje Z es el cual tomamos para considerar que el dispositivo se encuentra en movimiento. Para ello se estableció que los valores a tener considerar deben rondar entre Z > 10 u Z < -10. Estos valores deben ser constantes en un rango de tiempo no menor a 3sg, esto se establece para evitar falsas mediciones por movimientos normales en el uso del celular.

El rango de valores establecidos para Z se establecen en base a diferentes pruebas realizadas y además para despreciar el efecto de gravedad si el eje Z se encontrara perpendicular a la Línea de tierra.

Sensor Proximidad

Tanto la pantalla de tiempo real como el mapa, están suscriptas a los cambios del sensor:

- Pantalla tiempo real: si el usuario pasa su mano por delante del celular (como se ilustra en la imagen), provocará que se abra el mapa y podrá ver su ubicación actual.
- Pantalla mapa: si el usuario pasa su mano por delante del celular, esto provocará que el mapa se cierre.

Ventajas de uso:

La ventaja surge del siguiente análisis:

Supongamos que el usuario se encuentra manejando la bicicleta y quiere saber en dónde se encuentra (ubicación actual). Sin utilizar a este sensor, tendría que:

1. Prestar atención al celular para buscar al botón de mapa y presionarlo con la precisión que requiere el botón.
2. Mirar el mapa.
3. Continuar prestando atención al celular para buscar el botón de back y así cerrar el mapa.

Es decir, el mirar el celular conlleva esos tres tiempos. Es mucho teniendo en cuenta que si uno está manejando, debería bajar la vista al celular lo menos posible (solo hacer vistazos).

Gracias al sensor de proximidad pudimos reducir esos tres tiempos, a uno. Solamente el usuario mirará el celular para ver el mapa (que es lo que le interesa). Para abrirlo o cerrarlo, alcanza con que pase la mano por delante del mismo.

Filtros:

Solamente se considera el eje X. Este sensor normalmente tiene a su valor x como el valor más alto posible (maximum range). Si en algún momento su valor es menor, significa que tenemos un objeto cercano.

Sensor Giroscopio

La pantalla de Tiempo real se encuentra suscrita al cambio de los valores del sensor. Al detectar una variación en sus ejes, dentro del rango establecido, se ejecuta el diálogo de llamada de emergencia.

Se considera que si los diferentes valores devueltos por el sensor se encuentran dentro del rango establecido, los mismos representan una caída del ciclista o un evento brusco que puede significar realizar un llamado de emergencia

Es la combinación del osciloscopio y de los acelerómetros (sensores capaces de captar la inclinación del dispositivo y la aceleración de dicho movimiento) lo que permite al smartphone saber en qué plano del espacio se encuentra, cuál es su aceleración con respecto al punto de partida y si el dispositivo está girando sobre el plano o inclinándose.

Se toman variaciones en los ejes Y y Z para establecer el evento de movimiento brusco y posterior ejecución de la llamada de emergencia. Se desprecia el valor de X por la posición en la cual se va a encontrar el dispositivo cuando se encuentre en uso del sistema SBA. Los rangos de valores que se establecieron para dichos ejes fueron estimados por los resultados de las distintas pruebas realizadas.

Integración con Google Maps

Lo primero que debemos hacer es agregar el paquete de Google Play Services a Android Studio.

Luego, tenemos que generar una API_KEY para que nuestra aplicación pueda acceder a los servidores de Google Maps. A esta API_KEY debemos agregarla al AndroidManifest.xml de la siguiente manera:

<meta-data android:name="com.google.android.geo.API_KEY" android:value="YOUR_API_KEY"/>

Insertar esta línea dentro del elemento “application”.

Para poder ver al mapa, necesitamos crear un SupportMapFragment.

Por último, lo agregamos al layout de un FragmentActivity y ésta sería la pantalla que finalmente terminaría viendo el usuario.

Ubicación del usuario

EL motivo por el cual agregamos el mapa, es porque el usuario puede ver su ubicación en tiempo real.

Para poder lograr esto, primero, tenemos que solicitar permisos de localización al usuario. Luego, le pedimos al LocationManager la ubicación. Por último, hacemos zoom en la latitud y longitud correspondiente.

Lecciones aprendidas

Efecto rebote en medidas de sensores

Al intentar medir la salida de pulsadores o sensores digitales (como el Tilt Switch) notamos que las mediciones obtenidas en los cambios de estado no eran regulares por un corto intervalo de tiempo (orden de los milisegundos).

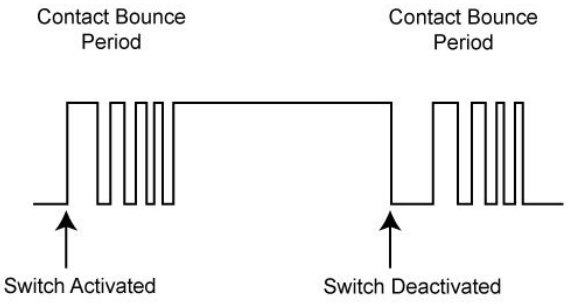


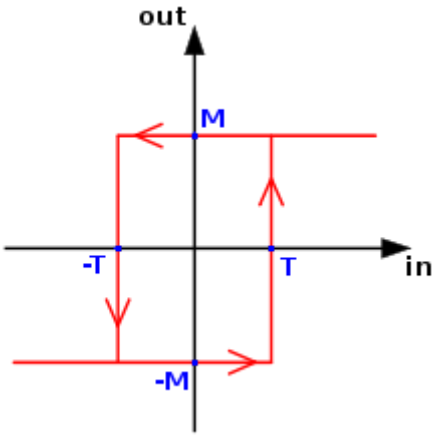
Gráfico de efecto rebote

Para sortear este efecto implementamos contadores sobre las lecturas de estos sensores. Básicamente, contamos una cantidad mínima de ciclos que deban transcurrir sobre un nivel (alto o bajo) para contabilizar el cambio de estado.

Respuestas más estables a cambios en el medio

Trabajando con el módulo de iluminación de chasis, al momento manejar la lógica de encendido y apagado de las luces LED nos topamos con que los cambios de estado podrían resultar oscilantes en los siguientes escenarios:

- Al estar circulando de día y cubrir el fotoresistor (LDR) por un momento (ej: sombra de un árbol) se activarían por ese breve instante las luces LED.
- En situaciones en las que la cantidad de luz estaba justo sobre el umbral que dividía el estado de enceder o apagar la iluminación.



Ciclo de histéresis de un sistema de control

Resolvimos este inconveniente separando los umbrales de encendido y apagado e incorporando retardos en los tiempos en que la entrada debería superar dichos umbrales. Los valores fueron ajustados en base a experimentación.

Básicamente, nos permite mantener un valor de salida estable hasta que el cambio en la entrada sea considerable en tiempo y magnitud.

Importancia de las funciones millis() y micros()

Nuestro sistema es de tiempo real. Esto quiere decir que los tiempos de respuesta son muy importantes.

Puede ocurrir que un sensor requiere una mínima cantidad de tiempo para poder trabajar. Esto se nos presentó con el sensor de ultrasonido. En un principio, habíamos utilizado la función delay() pasándole ese tiempo que requería el sensor. El gran problema que genera esta función es que estaba bloqueando a todo nuestro sistema. Esto no debe suceder. Para resolverlo, utilizamos la función micros(). Ésta nos devuelve el tiempo transcurrido desde que arrancó a ejecutarse el sketch, en microsegundos. La estrategia fue usar marcas de tiempo (timestamp) para guardar el momento en el que el sensor comenzó a trabajar. El loop del programa nunca se bloquea. Por cada loop, validamos si ya pasó el tiempo que requiere el sensor y si no, seguimos ejecutando el loop.

Tanto micros() como millis() sirven para resolver este tipo de problemas.

Futuras mejoras

- Exportar datos de sesión a un sistema web
  - Calendario de sesiones
  - Estadísticas: velocidad, distancia, tiempo de sesión promedio, etc
- Premios simbólicos por superar marcas personales
- Compartir perfil y resultados en redes sociales
- Mapa de ciclovías
  - Sugerencia de nuevas rutas, estadísticas de distancias recorridas.
- Configuraciones generales para el sistema (Ejem: Velocidad Máxima)

Casos de prueba

A continuación se establecen los casos de prueba a realizar sobre el prototipo con el objetivo de confirmar la consistencia del proyecto.
El objetivo es establecer mediante diferentes contextos de uso, la respuesta correcta del conjunto y su transición estable.

Contexto de prueba: Inicio de pruebas con bicicleta en estado Reposo.

Caso de Prueba 1					
Condición de operación	Prueba realizada	Resultado esperado	Observaciones	Cumple	No cumple
Modo Reposo	Simular objeto cercano detrás de la bicicleta	Sin funcionalidad objeto cercano			
	Accionar luz de giro	Sin funcionalidad Luz de Giro			
	Simular sin iluminación	Sin funcionalidad Sensor iluminacion			
	Simular movimiento brusco	Sin funcionalidad Alarma			

Contexto de prueba: Comienzo viaje, inició movimiento en la rueda trasera hasta superar los 5km/h.

Caso de Prueba 2					
Condición de operación	Prueba realizada	Resultado esperado	Observaciones	Cumple	No cumple
Modo Viaje	Verificar modo en App	Se verifica Inicio de viaje.			

	Accionar luz de giro	Activa Luz de Giro correspondiente			
	En movimiento verificar indicación velocidad en aplicación	Se visualiza la velocidad			
	Simular sin iluminación	Se activa iluminación chasis			
	Simular movimiento brusco	Sin funcionalidad Alarma activa. No se Activa Alarma Sonora y visual			
	Simular la aproximación de objeto detrás de la bicicleta	Se activa aviso sonoro de objeto cercano. Simultáneamente se visualiza en App el aviso			
	Pasar la mano por el frente del celular	Se visualiza el mapa en App			
	Detener el movimiento de la rueda por más de 60 segundos	Pasa a modo reposo			

Contexto de prueba: Finalizado el viaje activo alarma mediante llave manual o Aplicación Android (App)

Caso de Prueba 3					
Condición de operación	Prueba realizada	Resultado esperado	Observaciones	Cumple	No cumple
Modo Alarma Armada	Simular objeto cercano detrás de la bicicleta	Sin funcionalidad objeto cercano			
	Accionar luz de giro	Sin funcionalidad Luz de Giro			
	Simular sin iluminación	Sin funcionalidad Sensor iluminacion			
	Simular movimiento brusco (robo)	Se activa alarma Sonora y visual			
	Desactivar alarma mediante mensaje con App	Se desactiva alarma sonora y visual			
	Simular traslado de bicicleta con movimiento en la rueda (robo)	Se activa alarma Sonora y visual			
	Desactivar alarma mediante combinación de llave	Se desactiva alarma sonora y visual			

Contexto de prueba: Se activa la alarma y se genera por cualquier modo su activaciòn..

Caso de Prueba 4					
Condición de operación	Prueba realizada	Resultado esperado	Observaciones	Cumple	No cumple
Modo Alarma Sonando	Simular objeto cercano detrás de la bicicleta	Sin funcionalidad objeto cercano			
	Accionar luz de giro	Sin funcionalidad Luz de Giro			
	Simular sin iluminación	Sin funcionalidad Sensor iluminacion			
	Simular movimiento brusco (robo)	Sin cambio de estado. Alarma se mantiene activa.			
	Simular traslado de bicicleta con movimiento en la rueda (robo)	Sin cambio de estado. Alarma se mantiene activa.			
	Desactivar alarma mediante combinación de llave	Se desactiva alarma sonora y visual			