INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN SISTEMAS EMBEBIDOS

DEBI (DEVICE FOR BIKES)

Tesina para obtener el grado de:

ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presentan: Emmanuel de Jesús Santiago Saavedra y Felipe de Jesús Mercado Castellanos

Director: Mtro. Carlos Arturo García Camarena

San Pedro Tlaquepaque, Jalisco. Mayo de 2018.

Acknowledgments

We thank God, our families, teachers, ITESO, and CONACYT (Scholarships IDs: 862289 and 860914), for helping us achieve one more step towards our professional aspirations and goals.

Abstract

In the city of Guadalajara, Jalisco in Mexico, the safety of cyclists in urban areas is a current problem. The government is investing in infrastructure like bicycle paths and other mobility programs to solve this issue, but there is still a lack of security measures that allow cyclists to ride safely, for example, the incorporation of technological devices with safety features. DEBI (Device for Bikes) is a product developed as a proof of concept to provide active safety features for cyclists in urban areas by implementing a low-cost and portable device. The main features of DEBI include: LED lights used to indicate when the cyclist is stopping and turn signals that are controlled via wireless using a software application running on a smartphone. DEBI can also detect when other vehicles are approaching in collision course and activate an acoustic alarm to alert the cyclist. All these features are achieved thanks to accelerometer with Microelectromechanical Systems technology (MEMs) and an ultrasonic proximity sensor that communicates with the core component, the ATmega328p microcontroller by Atmel™, integrated in the Bluno Beetle™ development board with a built-in Bluetooth communication module. The result was satisfactory considering the scope of this project and all safety features were validated. However, it is important to note that issues related to false-positive detection were found under specific circumstances, this could be corrected in future versions using a more robust proximity sensor, but it will require further analysis since this could significantly increase the cost of the device.

Resumen

En la ciudad de Guadalajara, Jalisco en México, la seguridad de los ciclistas en áreas urbanas es un problema actual. El gobierno está invirtiendo en infraestructura como ciclovías y otros programas de movilidad para resolver este problema, pero aún existe escases de dispositivos con funcionalidad de seguridad para ciclistas- El dispositivo llamado DEBI (device for bikes) fue desarrollado como prueba de concepto que proporciona características de seguridad activa para ciclistas en áreas urbanas con la implementación de un dispositivo económico y portable. Las principales características de DEBI incluyen: Luces LED que son usadas para indicar cuando el ciclista está frenando y luces direccionales que son controladas de forma inalámbrica usando una aplicación de software ejecutada desde un smartphone. DEBI puede también detectar cuando otros vehículos se acercan en curso de colisión y activa una alarma acústica para alertar al ciclista. Todas estas características son alcanzadas gracias al acelerómetro con tecnología MEMs (Microelectromechanical Systems) y el sensor ultrasónico de proximidad que se comunica con el componente principal que es el microcontrolador ATmega328p de Atmel que está integrado en la tarjeta de desarrollo Bluno Beetle que además cuenta con un módulo de comunicación bluetooth embebido. Los resultados fueron satisfactorios considerando el alcance de este proyecto y todas las funcionalidades de seguridad fueron probadas. Sin embargo, es importante notar que algunas cuestiones relacionadas con lecturas falso-positivas fueron encontradas bajo circunstancias específicas, esto podría ser corregido en futuras versiones usando un sensor de proximidad más robusto, aunque esto requiere un análisis más detallado debido a que incrementaría de manera significativa el costo del dispositivo.

List of Figures

List of Tables

Table of contents

Introduction

DEBI ['de βi] stands for Device for Bikes and it arises as a safety solution for bike riders in urban areas during traffic hours. This device is a conjunction of hardware and software modules that work together to achieve the goal of data acquisition related to bike riding like acceleration and proximity of other vehicles on the road. It also has visual indicators (brake and tail lights) for helping other vehicles become aware of the presence of the bike rider as well as informing the rider the presence of a vehicle that is getting closer.

In the city of Guadalajara, the government has invested in mobility programs to increase non-motorized transportation with the intention of reducing the pollution and traffic as well as promoting physical activity among the citizens and MIBICI is one of those programs [1]. The project DEBI was created as a result of the urban mobility problem in Guadalajara, where the increased number of cars reduces the mobility considerably and causes stress in the population. Consequently, many citizens are switching to alternative transportation methods and the bicycle is one of the most popular and 98% of the people are able to use it [2].

The city of Guadalajara located in the state of Jalisco has one of the largest vehicular fleets of Mexico. In fact, Jalisco has one of the highest growing vehicle registration rates according to National Institute of Geography and Statistics (INEGI) [3]. Considering this growing rate, people need to search for alternative transportation, such as a bicycle. One of the main issues encountered by cyclists in urban areas is safety, bicycle riders are faced with high risk because of the lack of driver education found in some Mexican cities like Guadalajara and therefore, riding a bicycle in the Guadalajara Metropolitan Area could represent a potential risk for cyclist collision injuries, including permanent injuries, or even death.

DEBI is a safety bicycle accessory to be used while riding a bicycle in urban areas. The main goal of DEBI is cyclist safety, it has two built-in sensors to obtain information from the environment and pass it through the microcontroller that is based on an algorithm that can take actions to prevent any possible accident. One of the sensors is an accelerometer that is continuously sensing acceleration to identify when the bike is braking and turn on the stop signal, similar to what a car does, and thus, inform other drivers that the bicycle is stopping. The second built-in sensor detects the proximity, which is in charge of sensing the distance between the bike and other vehicles that are approaching. If the algorithm embedded in DEBI detects that the vehicle getting closer is in a collision course, an optic and acoustic alarm is raised to inform both, the bike rider and other vehicles.

Another important feature of DEBI is the capability of being used as a directional light to indicate other vehicles when the bike rider is turning left or right, and this is achieved by voice commands using the headphones of a smartphone equipped with an operating system Android® that is connected to DEBI over the wireless communication protocol Bluetooth.

From a technical perspective, the intention of DEBI is to provide a low-cost, adaptable, and useful device that bike riders can use no matter their age or social stratum. DEBI is designed using inexpensive and relatively common components that built together are transformed in a unique portable device that has a rechargeable battery and can be mounted on a bicycle quickly and with no need of special knowledge in electronics or mechanics.

Therefore, the main goal of this project is the creation of a proof of concept, in other words, our objective is to demonstrate that it is possible to take advantage of an existing technology and components to build an innovative, price-competitive, and adaptable device that will improve safety while riding a bicycle in urban areas, which is a relevant issue in today’s society.

1. Background

One of the leading countries regarding the development of bicycle gadgets that focuses on cyclist safety is the United States. In most cases, the main goal is to achieve visibility on the road and offer visual indicators to warn drivers about cyclist presence. For instance, “Lucid Brake” [4] is a simple accessory for bicycles that is powered by non-rechargeable batteries; it works as a stop light with different blinking patterns that can be easily attached to the bike. Another device called “Blinkers” [5] is more complex than “Lucid Brake” because in addition to the stop light, it also incorporates tail lights to inform other drivers the cyclist’s intention of turning left or right. Blinkers has a built-in rechargeable battery and a light sensor that allows the automatic calibration of the light intensity.

On the other hand, there are devices that also implement sensors to detect the proximity of other vehicles and trigger an alarm to prevent a possible collision. The most popular device of this type is presented by the multinational company Garmin with “Varia radar” [6], which is a security radar for highway cyclists. There is also another gadget known as “Bikesphere” [7], which has a lower proximity detection range but an additional feature: this device projects a circle on the floor using a laser beam to delimit a secure zone.

In Mexico, similar gadgets with these safety features are not commonly available. Nonetheless, local institutions have shown their concern about urban mobility in a safe and efficient way. Recently in 2014, the Instituto de Movilidad y Transporte del Estado de Jalisco (IMTJ) has implemented a public bicycle service oriented towards a sustainable mobility [8]. IMJT is responsible for promoting urban mobility and sustainable transportation through the development and execution of projects, which includes design and research regarding urban mobility in the state of Jalisco.

MIBICI is a public transportation program implemented in the metropolitan area of Guadalajara in the state of Jalisco. MIBICI was designed to improve urban mobility through bicycle rental services. Available bicycles are located in stations across the city in strategic points where users can pick a bicycle in a station and once the trip has been completed, it can be returned in any of the available stations.

The MIBICI project is based on Jalisco’s State Development Plan (2013-2033) [9]. In this plan, theOD603 objective highlights the need of having collective and massive alternative transportation systems that meet quality, safety, and efficiency standards. In this sense, DEBI is introduced as a solution that incorporates the most relevant features of the devices that are currently in the market by implementing them in an efficient way, specifically, it integrates proximity and acceleration sensors that are continuously acquiring data from the environment. This data is then processed by a microcontroller that can perform actions depending on the current situation, such as informing the cyclist about possible collision threats with vehicles, via an acoustic alarm, and warning other drivers by switching on the rear lights.

2. Theoretical framework

As an embedded system, this project is a conjunction of software, hardware, and a physical component that contains the system [10]. DEBI is constituted by hardware (sensors, a microcontroller, a communication module, and visual indicators), software (the program that processes the data and takes decisions) and it is embedded in a small rectangular box that allows the portability of the device. Figure 2-1 shows in what manner the components are related as well as the inputs and outputs.

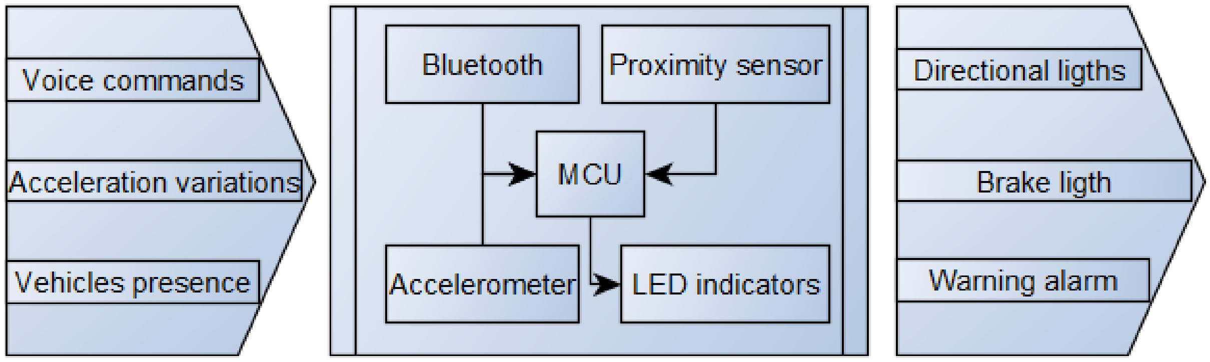


Figure 2-1. DEBI Boundary Diagram.

2.1. Microcontroller

The core component of DEBI is a very popular microcontroller, the Atmega328p. This is a low power 8-bit AVR family RISC (Reduced Instruction Set Computer) microcontroller whose most relevant characteristics are: a 32KB ISP (In System Programming) flash memory with read-while-write capabilities, 1024B EEPROM (Electrically Erasable Programmable Read-Only Memory), 2KB SRAM (Static Read-Only Memory) with internal and external interrupts, serial programmable USART (Universal Serial Asynchronous Receiver-Transmitter), SPI (Serial Peripheral Interface), and a 6-channel 10-bit resolution ADC (Analog to Digital Converter) [11].

As it has been mentioned before, one of the main objectives of DEBI is to be a low cost device, for this reason, the proof of concept was built using a development board based on Arduino, the most popular open source hardware & software platform around the world [12]. The Bluno Beetle development board was selected for this project, which incorporates the Atmega328p microcontroller as well as a Bluetooth module that allows a wireless communication, both encapsulated in a tiny board of around one square inch [13]. In Table 2-1, the most relevant specifications of Bluno Beetle are described.

Table 2-1. Bluno Beetle main specifications.

Parameter	**Value**
Microcontroller	ATmega328P
Working Temperature	(-10 ℃ ~ +85 ℃)
Clock frequency	16 MHz
Working voltage	5V DC
Bluetooth Chip	CC2540
Maximum Distance	50 m (Open field)
Sensitivity	(-93dBm)
Digital Pin	x4
Analog Pin	x4
PWM Output	x2
UART interface	x1
I2C interface	x1
Micro USB interface	x1

This development board has a very simple design and size that combines a powerful microcontroller and the wireless communication module embedded in the same board (Figure 2-2).

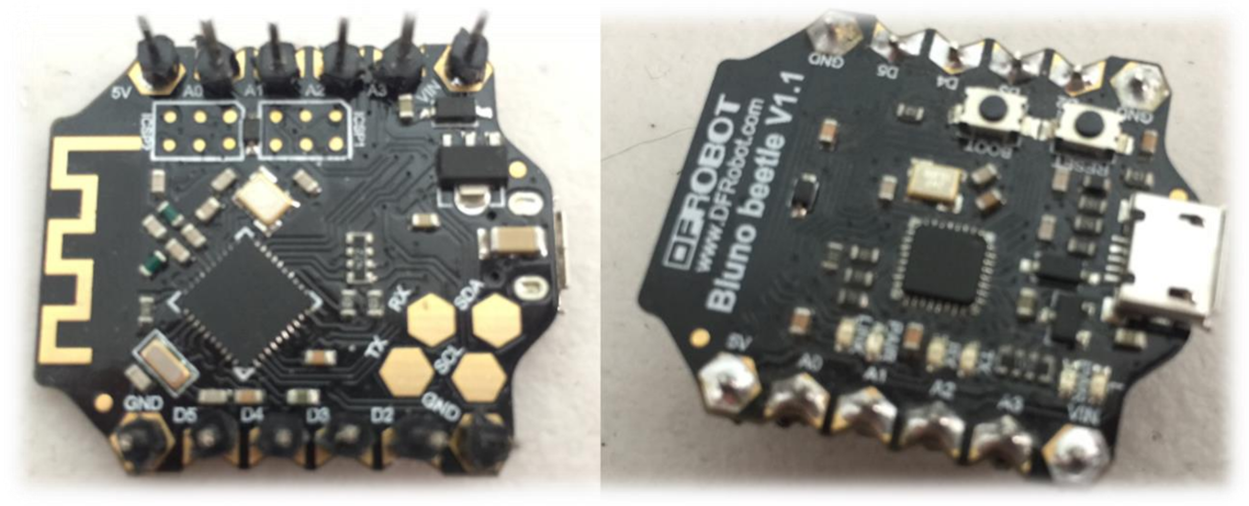


Figure 2-2. Bluno beetle v1.1.

2.2. Sensors

Sensors are a crucial part of this project. In total, two sensors were used in DEBI, one of them is an accelerometer that is in charge of sensing the changes in acceleration in order to detect when the bike is slowing down and the other is a proximity sensor that is continuously measuring the distance between rear vehicles and the bike.

2.2.1 Acceleration sensor

As mentioned before, the accelerometer is used by DEBI to detect when the vehicle is stopping and a sensor based on micro-electro-mechanical systems (MEMS) technology was selected due to the low-cost and high performance of these modules. The module used was the ADXL335, it is a 3-axis accelerometer with a signal conditioned voltage output and the analog output where the range of measurement is ±3g [14] as shown in Table 2-2.

Table 2-2. Accelerometer ADXL335 specifications.

Parameter	Value
Model	ADXL335
Operating voltage range	1.8 to 3.6 V
Operating temperature range	-40 to +85 °C
Number of axis	3
Measurement range	±3g
Nonlinearity	±0.3% of full scale
Output voltage range	0 to 3.3 V

2.2.2 Proximity sensor

Proximity sensors are classified by the physical magnitude that is used to calculate the relative position of an object; inductive and capacitive sensors use the electromagnetic field, photoelectric sensors use electromagnetic waves, for instance the light, and ultrasonic sensors use mechanical waves like the sound [15]. All of them are suitable for different applications because of their advantages and restrictions. After comparing them, a proximity sensor based on the ultrasonic technology was selected because it offers the best characteristics for this application. These characteristics are shown below:

- Small in size and easy to use
- High precision for measurements
- Anti-interferences capabilities

The model of the ultrasonic sensor used for this project is the JSN-SR04T v 2.0 (Figure 2-3). Aside from the characteristics mentioned above, this module incorporates a waterproof sensor that is very useful for outdoor applications.

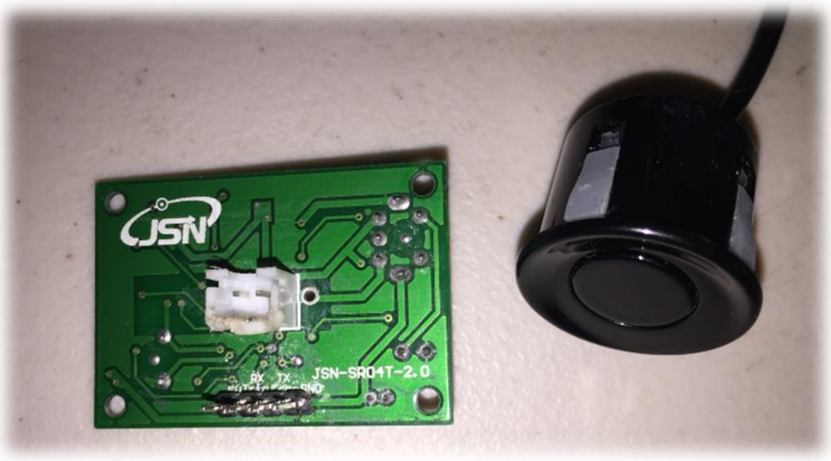


Figure 2-3. Ultrasonic sensor JSN-SR04T v2.0.

The main specifications of this sensor are listed in Table 2-3. This sensor has an operational voltage of 5V and the detection range goes from .25 to 4 meters, which is enough for this application and the detection angle of 70° meets the requirements.

Table 2-3. JSN-SR04t v2.0 specifications

Parameter	**Value**
Operating voltage	5 V DC
Static current	5 mA
Working current	40 mA
Working Range	0.25 to 4m
Resolution	3 mm
Detection angle	70°
Working temperature	-10 ~ 70°C

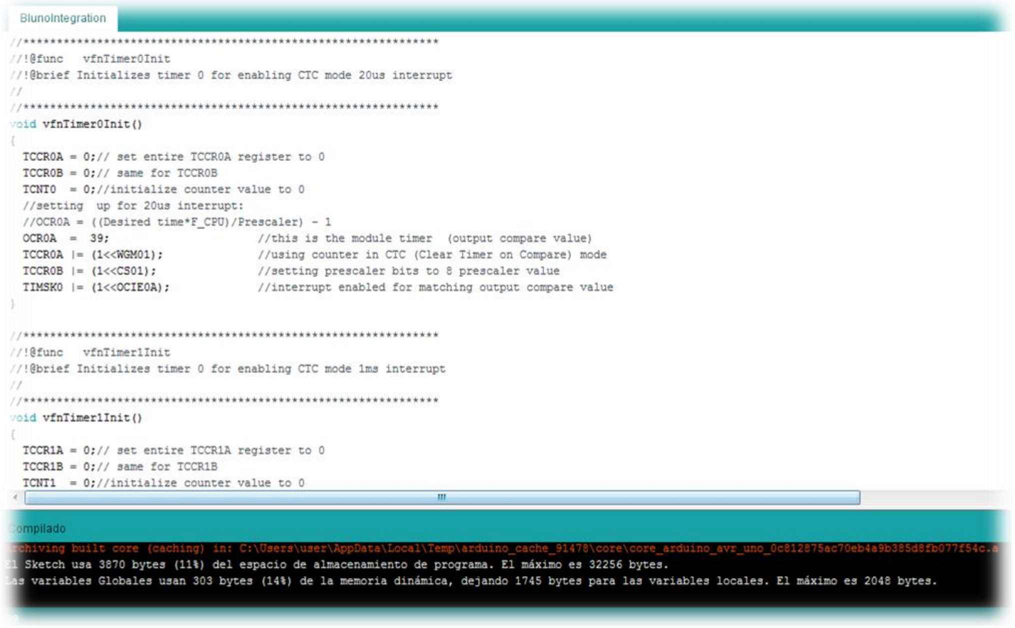
2.1. DEBI Software

2.1.1 Language

DEBI's software is composed of two software applications, an embedded software application running into the Bluno Beetle (™) board and a smartphone application running into an Android OS (™) device.

Bluno Beetle incorporates an Atmega328 (™) MCU (Microcontroller Unit), which is used as its main processor unit. This microcontroller may be programmed with a PC (Personal Computer) software, specifically this software is called an IDE (Integrated Development Environment). Since the Bluno Beetle board is going to be used as the development platform, the most appropriate IDE for programming is the Arduino IDE (™) (see Figure 2-4). This IDE allows to load firmware in the Atmega328p microcontroller flash memory embedded in a Bluno Beetle board through an USB (Universal Serial Bus) wire connected to the PC. This IDE supports both C and C++ programming languages. DEBI's source code is written using C programming language.

There is another IDE called Atmel Studio, which incorporates the possibility of running a debug session in conjunction with several hardware tools, such as Atmel ICE (™) and AVR Dragon (™) that allow detecting and fixing software bugs (code issues) by running the code step by step or allowing to set breakpoints for stopping the executed program at a certain point in time. Atmel studio also supports C and C++ programming languages. As supporting software and hardware resources for solving code bugs, Atmel Studio (™), AVR Dragon (™), and a standalone Atmega328(™) MCU programmer were considered. This IDE is used for code compiling and firmware load processes to Bluno Beetle board. (Figure 2-4)


BlunoIntegration

```
//****************************************************************
//!@func   vfnTimer0Init
//!@brief Initializes timer 0 for enabling CTC mode 20us interrupt
//
//****************************************************************
void vfnTimer0Init()
{
  TCCR0A = 0;// set entire TCCR0A register to 0
  TCCR0B = 0;// same for TCCR0B
  TCNT0  = 0;//initialize counter value to 0
  //setting  up for 20us interrupt:
  //OCR0A = ((Desired time*F_CPU)/Prescaler) - 1
  OCR0A  =  39;                   //this is the module timer  (output compare value)
  TCCR0A |= (1<<WGM01);           //using counter in CTC (Clear Timer on Compare) mode
  TCCR0B |= (1<<CS01);            //setting prescaler bits to 8 prescaler value
  TIMSK0 |= (1<<OCIE0A);          //interrupt enabled for matching output compare value
}

//****************************************************************
//!@func   vfnTimer1Init
//!@brief Initializes timer 0 for enabling CTC mode 1ms interrupt
//
//****************************************************************
void vfnTimer1Init()
{
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
```

ompilado

```
rchiving built core (caching) in: C:\Users\user\AppData\Local\Temp\arduino_cache_91478\core\core_arduino_avr_uno_0c812875ac70eb4a9b385d8fb077f54c.a
l Sketch usa 3870 bytes (11%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
as variables Globales usan 303 bytes (14%) de la memoria dinámica, dejando 1745 bytes para las variables locales. El máximo es 2048 bytes.
```

Figure 2-4. Arduino IDE.

Finally, a smartphone application compatible with Android OS (Operating System) is considered as part of DEBI’s user control interface for activating directional lights. This software application is developed using MIT App Inventor, which is a visual programming environment available online. [16]

2.2. DEBI Physical design

DEBI measurements are 13.5 x 4.9 x 7.5 cm, it is made of resistant plastic and all the electronics are mounted inside to create a portable device as shown in Figure 2-5.

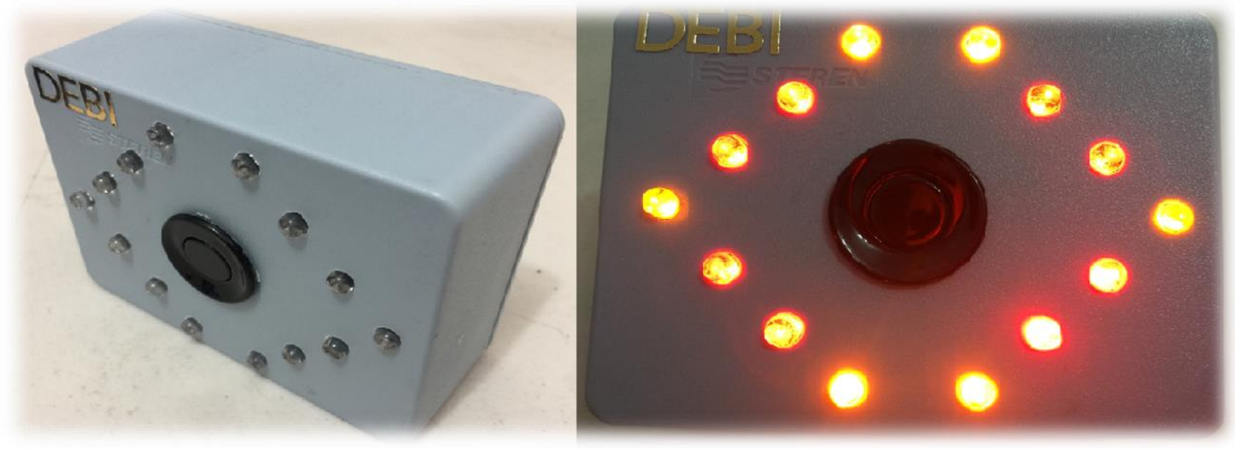


Figure 2-5. DEBI Physical design.

2.3. DEBI Principle of operation

It has been already mentioned that DEBI is a portable device with safety features for cyclists. Figure 2-6 shows how DEBI is intended to be used in a real-life situation. DEBI is placed on the rear part of the bicycle with the lights pointing to the back where the ultrasonic sensor is continually sensing the environment to detect when a vehicle is approaching too fast and thus, raise the alarm.

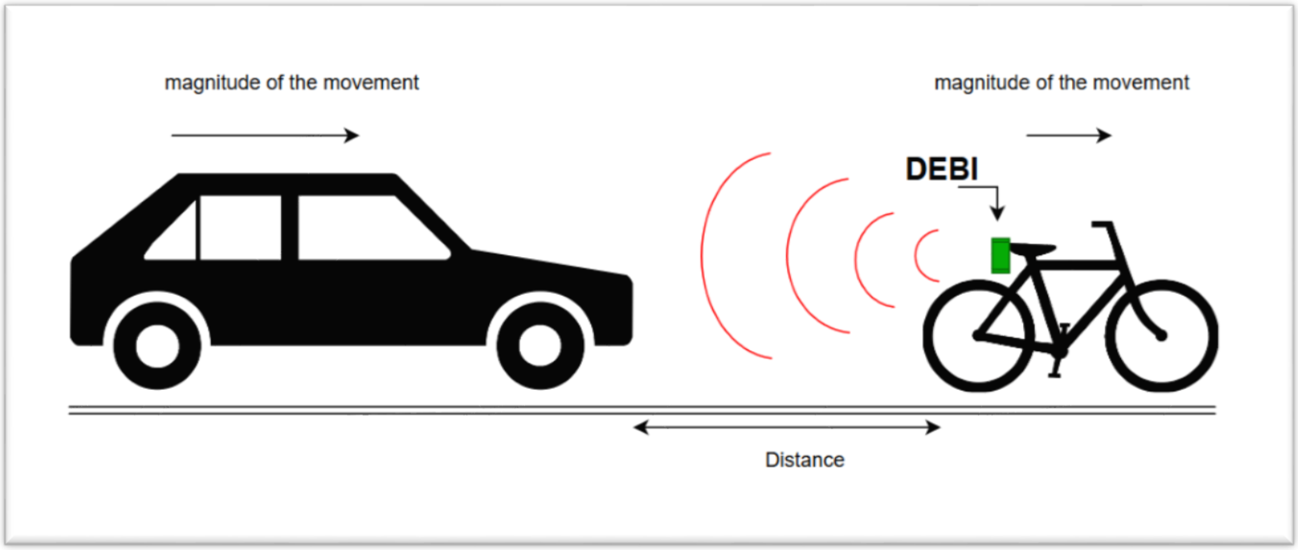


Figure 2-6. DEBI Principle of operation.

3. Methodology

For the realization of this project an inexpensive, accessible, and with enough capabilities microcontroller was selected to achieve DEBI’s objective: the Atmega328p from Atmel.

MCU peripherals needed for achieving all DEBI capabilities are: ADC (Analog to Digital Converter), UART (Universal Asynchronous Receiver-Transmitter), GPIO's (General Purpose Inputs-Outputs), and Timers, all of these peripherals are available in Atmega328p MCU.

3.1. MCU peripherals

3.1.1 UART (Universal Asynchronous Receiver-Transmitter)

For DEBI, Atmega328p receiver (Rx) functionality is used for listening to incoming messages from a smartphone application, just as turn left or right commands from the user. Transmitter (Tx) functionality is considered for further versions and it will be used for sending alerts to the smartphone application for notifying the user about a detected threatening situation (risk of vehicle collision).

3.1.1.1 UART Settings

UART peripheral is configured as 115200 bauds, 8 data bits, non-parity bit, and 1 stop bit.

3.1.1.2 UART Algorithm

UART is handled through a polling method, which means, data received register is periodically polled for checking if new data has arrived. Receiver algorithm is shown in Figure 3-1.

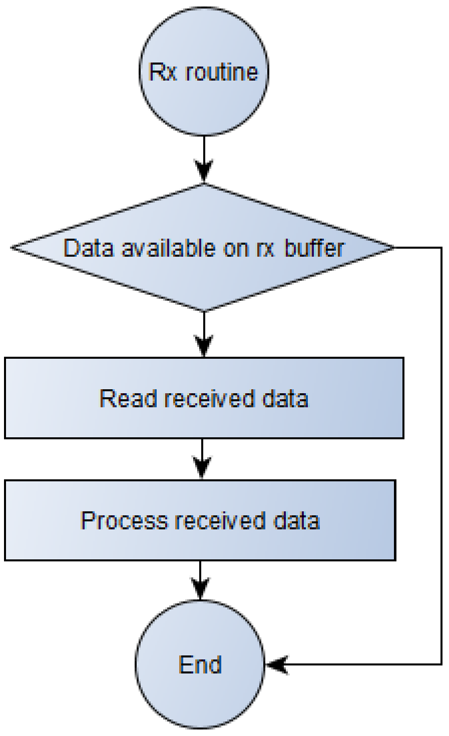


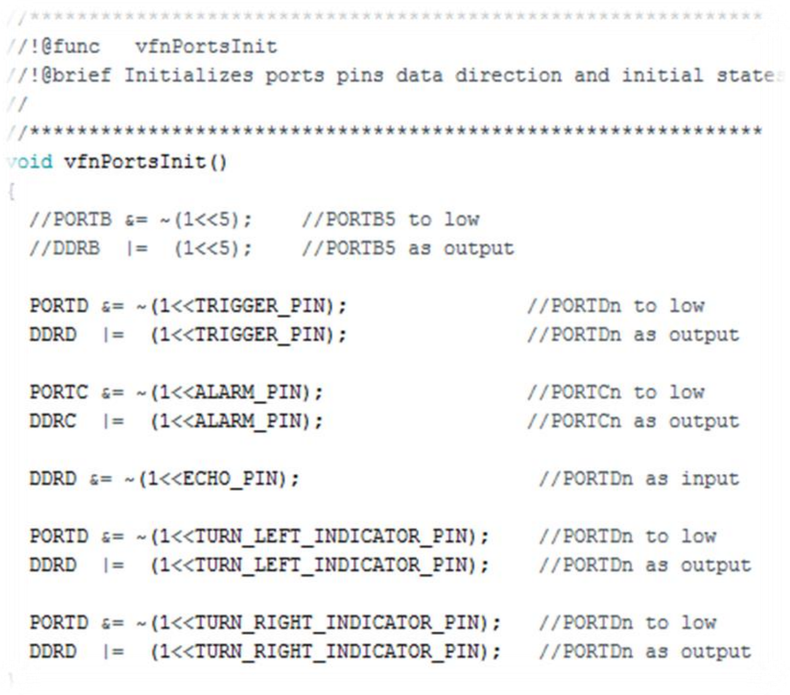
Figure 3-1. Receiver routine algorithm. Rx routine is handled through polling method.

3.1.2 GPIO's (General Purpose Inputs-Outputs)

GPIO's are used for handling all DEBI's hardware modules: accelerometer, turn left and turn right indicators, buzzer alarm, and ultrasonic sensor. GPIO's functionalities are listed in Table 3-1 and the source code for the initialization is shown in Figure 3-2.

Table 3-1. Bluno Beetle Board™ -Atmega328p pins relationship and functionality.

Bluno Beetle pinout	Atmega328p pin	Functionality
D2	PD2	Output: Turn Left Indicator
D3	PD3	Output: Turn Right Indicator
D4	PD4	Input: Scan pulse width proportional to object (vehicle) distance
D5	PD5	Output: Start object distance detection
A3	PC3	Output: Audible alarm (buzzer)
A2	PC2	Input: Accelerometer X axis readings
A1	PC1	Input: Accelerometer Y axis readings
A0	PC0	Input: Accelerometer Z axis readings
TX	PD1	UART Transmitter for sending data to CC2540 Bluetooth chip
RX	PD0	UART Receiver for receiving data from CC2540 Bluetooth chip



```
//**************************************************************
//!@func   vfnPortsInit
//!@brief Initializes ports pins data direction and initial state
//
//**************************************************************
void vfnPortsInit()
{
  //PORTB &= ~(1<<5);    //PORTB5 to low
  //DDRB  |=  (1<<5);    //PORTB5 as output

  PORTD &= ~(1<<TRIGGER_PIN);                //PORTDn to low
  DDRD  |=  (1<<TRIGGER_PIN);                //PORTDn as output

  PORTC &= ~(1<<ALARM_PIN);                  //PORTCn to low
  DDRC  |=  (1<<ALARM_PIN);                  //PORTCn as output

  DDRD &= ~(1<<ECHO_PIN);                     //PORTDn as input

  PORTD &= ~(1<<TURN_LEFT_INDICATOR_PIN);     //PORTDn to low
  DDRD  |=  (1<<TURN_LEFT_INDICATOR_PIN);     //PORTDn as output

  PORTD &= ~(1<<TURN_RIGHT_INDICATOR_PIN);    //PORTDn to low
  DDRD  |=  (1<<TURN_RIGHT_INDICATOR_PIN);    //PORTDn as output
}
```

Figure 3-2. Ports initialization code.

From Figure 3-2, note that UART and ADC peripherals pins are not initialized here, instead, they are initialized in their corresponding peripheral initialization functions.

3.1.3 ADC (Analog to Digital Converter)

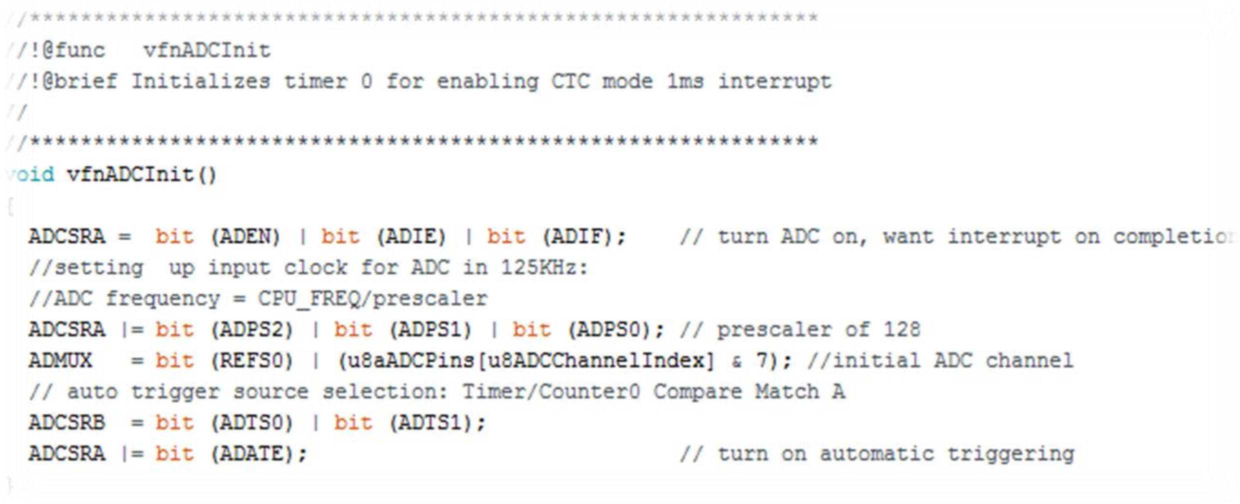
ADC peripheral is used to get data readings from the accelerometer, specifically X, Y, and Z axis values (acceleration in g forces units). Atmega328p MCU incorporates an 8 channel, 10bits resolution ADC peripheral.

3.1.3.1 ADC Settings

ADC peripheral performs conversions from analog readings to digital values at a specific periodical interval, this periodicity is limited to a range between 50KHz and 200KHz for Atmega328p [11] in a 10 bits resolution format, since CPU frequency is set at 16MHz, it needs to be prescaled, then, a value between the described range is achieved; this is accomplished by prescaling (dividing) the clock source as described in Equation (3-1). Prescaler values of 2, 4, 8, 16, 32, 64, and 128 are available for this peripheral. For DEBI, a prescaler value of 128 is chosen so the ADC peripheral works at 125KHz.

$$ADC\ frequency = \frac{CPU\ Frequency}{Prescaler\ Value} \tag{3-1}$$

ADC is set for working in auto trigger mode, which means the ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected to be timer/counter 0 compare match A, in other words ADC will start a conversion every rising edge of timer/counter 0 compare match A interrupt flag (Figure 3-3).

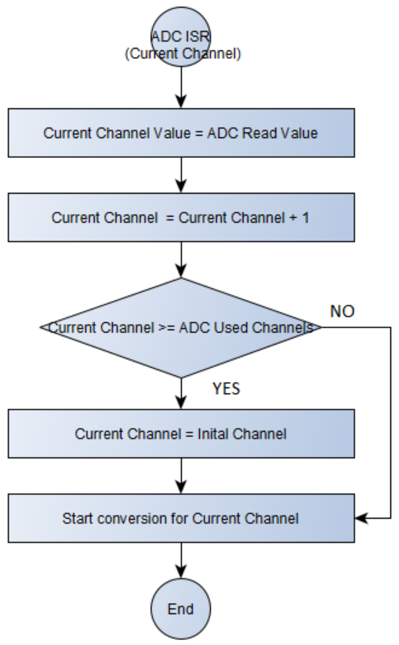


```
//******************************************************************
//!@func   vfnADCInit
//!@brief Initializes timer 0 for enabling CTC mode 1ms interrupt
//
//******************************************************************
void vfnADCInit()
{
  ADCSRA =  bit (ADEN) | bit (ADIE) | bit (ADIF);    // turn ADC on, want interrupt on completio
  //setting  up input clock for ADC in 125KHz:
  //ADC frequency = CPU_FREQ/prescaler
  ADCSRA |= bit (ADPS2) | bit (ADPS1) | bit (ADPS0); // prescaler of 128
  ADMUX   = bit (REFS0) | (u8aADCPins[u8ADCChannelIndex] & 7); //initial ADC channel
  // auto trigger source selection: Timer/Counter0 Compare Match A
  ADCSRB  = bit (ADTS0) | bit (ADTS1);
  ADCSRA |= bit (ADATE);                             // turn on automatic triggering
}
```

Figure 3-3. ADC peripheral initialization routine.

3.1.3.2 ADC Algorithm

ADC is handled by enabling end of conversion interrupt. ISR (Interrupt Service Routine) algorithm is shown in Figure 3-4.



ADC ISR (Current Channel)
Current Channel Value = ADC Read Value
Current Channel = Current Channel + 1
Current Channel >= ADC Used Channels
NO
YES
Current Channel = Inital Channel
Start conversion for Current Channel
End

Figure 3-4. ADC end of conversion (ISR) algorithm.

As it can be noticed from Figure 3-4, at the end of conversion of each ADC channel, a conversion starts for the next ADC channel used. As mentioned above, three ADC channels are used for DEBI (for handling X, Y, and Z axis of the accelerometer).

3.1.4 Timers

Timers on DEBI are used for controlling the pace of the system, delays, and time length of operations, such as a delta (difference) time between two consecutive operations, for example, calculating how much time it takes to obtain two consecutive data acquisitions as proximity sensing through an ultrasonic sensor. In this case, the time is calculated based on systems ticks performed within the routines of each timer.

3.1.4.1 Timer settings

Two Atmega328p timers are used, timer 0 and timer 1. Timer 0 is an 8-bit timer module with an output compare module, that is to say, the output compare register may be used for comparing current timer count value and whenever the output compare register value equals the timer counter register value, then, the comparator signals a match. This event will set a flag that may be used for triggering an output compare interrupt. For timer 1 the principle of operation is the same than timer 0 with the difference that timer 1 a 16-bit register.

For DEBI, both timers (timer 0 and timer 1) work in output compare interrupt mode so, timer overflow for these interrupts shall be calculated to reach an appropriate value according the requirements of each timer. Timer 0 is chosen for generating the pulse needed for proximity sensor to trigger the sensing process. For this purpose, a timer interrupt with a periodicity of 10 microseconds is needed, nevertheless, a periodical 10 microsecond interrupt could represent an important CPU (Central Process Unit) load for the microcontroller, which could cause a big impact in other tasks, such as increasing speed and distance calculations. Experimentally, it was demonstrated that with a pulse of 20 microseconds accurate readings were obtained from this sensor, which reduced the CPU load, then a 20-microsecons periodical timer interrupt is chosen for timer 0; calculations for obtaining such value for this timer interrupt are described in Equation (3-2) to Equation (3-6).

From Atmega328p datasheet we have a formula (Equation 3-2) for a period of 50% duty cycle waveform.

$$fOCnA = \frac{fclock}{2N(1 + OCRnA)} \tag{3-2}$$

From Equation 3-2 fOCnA is the timer interrupt occurrence frequency, $fclock$ is the CPU clock frequency, N is the selected prescaler value for CPU clock, and n is the selected timer module (channel).

Isolating for OCRnA and considering output compare interrupt occurrence (half of a period):

$$OCRnA = \left(\frac{fclock}{(N)(fOCnA)}\right) - 1 \tag{3-3}$$

In terms of time:

$$OCRnA = \left(\frac{TOCnA}{(N)(Tclock)}\right) - 1 \tag{3-4}$$

From Equation (3-4) TOCnA is the timer interrupt occurrence period, and Tclock is the CPU clock period.

As mentioned before, CPU frequency is 16MHz then Tclock is:

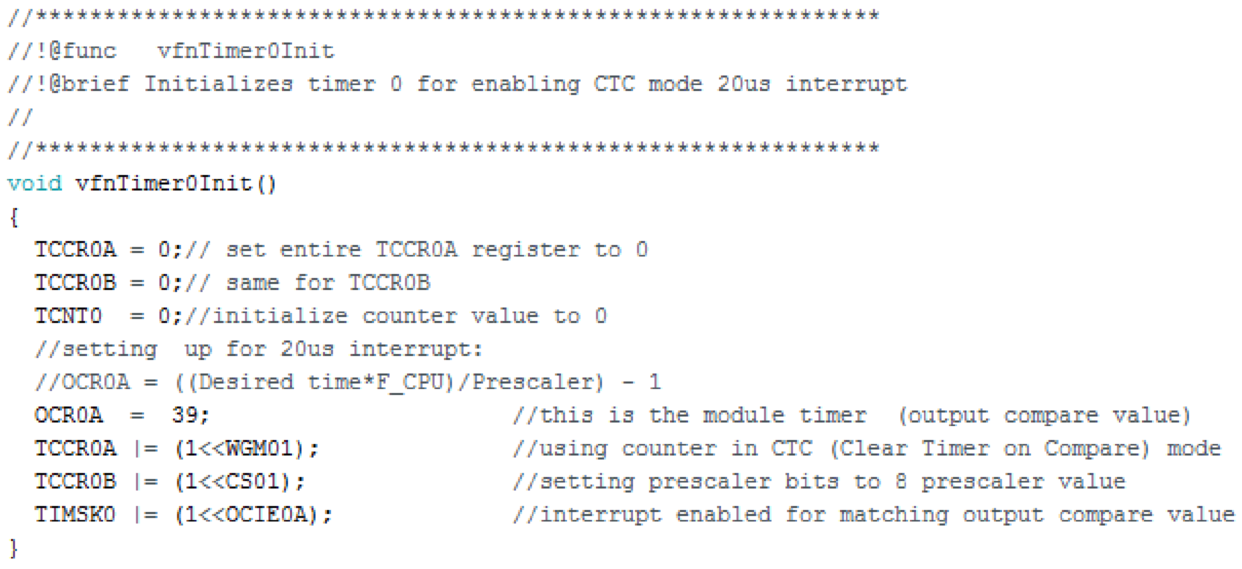
$$Tclock = \left(\frac{1}{16MHz}\right) = 62.5x10^{-9}s \tag{3-5}$$

For timer 0 TOC0A is 20us, so OCR0A value is calculated as shown in Equation 3-6:

$$OCR0A = \left(\frac{20x10^{-6}s}{(8)(62.5x10^{-9}s)}\right) - 1 = 39 \qquad (3\text{-}6)$$

Source code for timer 0 peripheral initialization is shown in Figure 3-5.

.



```
//****************************************************************
//!@func   vfnTimer0Init
//!@brief Initializes timer 0 for enabling CTC mode 20us interrupt
//
//****************************************************************
void vfnTimer0Init()
{
  TCCR0A = 0;// set entire TCCR0A register to 0
  TCCR0B = 0;// same for TCCR0B
  TCNT0  = 0;//initialize counter value to 0
  //setting  up for 20us interrupt:
  //OCR0A = ((Desired time*F_CPU)/Prescaler) - 1
  OCR0A  =  39;                    //this is the module timer  (output compare value)
  TCCR0A |= (1<<WGM01);            //using counter in CTC (Clear Timer on Compare) mode
  TCCR0B |= (1<<CS01);             //setting prescaler bits to 8 prescaler value
  TIMSK0 |= (1<<OCIE0A);           //interrupt enabled for matching output compare value
}
```

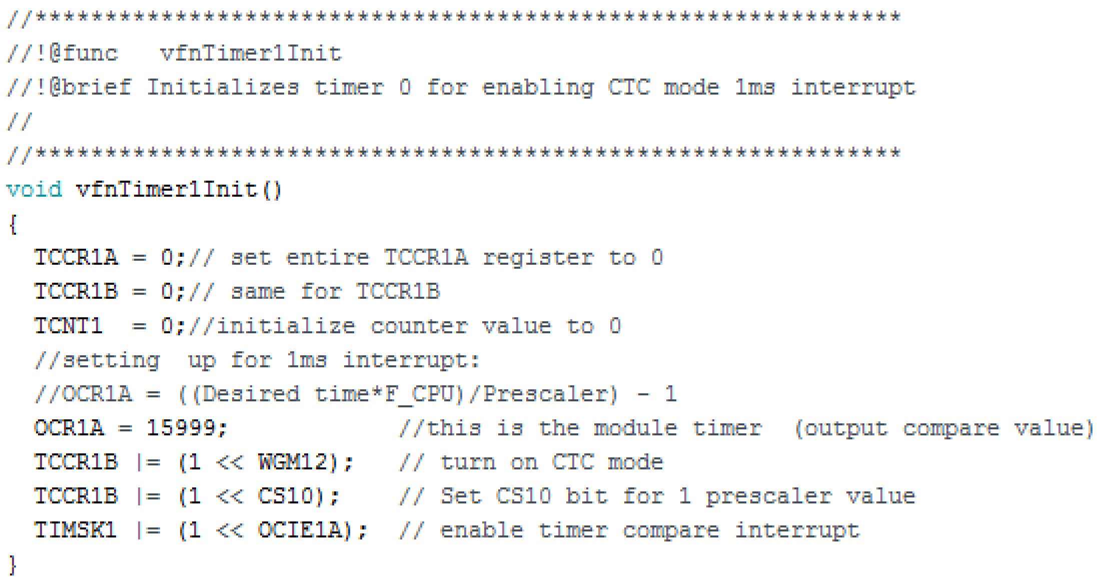
Figure 3-5. Timer 0 peripheral initialization routine.

Timer 1 is used for system control of pace, in other words, 1 millisecond periodical interrupt allows different software counters to be incremented on every timer interrupt, then, multiplicity of 1 millisecond values could be reached, thus, different time delay values could be generated, and also, elapsed periods between tasks could be calculated. Equation 3-7 shows the calculations for obtaining 1 millisecond periodical interrupt on timer 1.

For timer 1 $TOC1A$ is 1ms, so $OCR1A$ value is calculated as shown in Equation (3-7).

$$OCR1A = \left(\frac{1x10^{-3}s}{(1)62.5x10^{-9}s}\right) - 1 = 15999 \quad (3\text{-}7)$$

Source code for timer 1 peripheral initialization is shown in Figure 3-6.



```
//****************************************************************
//!@func   vfnTimer1Init
//!@brief Initializes timer 0 for enabling CTC mode 1ms interrupt
//
//****************************************************************
void vfnTimer1Init()
{
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  //setting  up for 1ms interrupt:
  //OCR1A = ((Desired time*F_CPU)/Prescaler) - 1
  OCR1A = 15999;            //this is the module timer  (output compare value)
  TCCR1B |= (1 << WGM12);   // turn on CTC mode
  TCCR1B |= (1 << CS10);    // Set CS10 bit for 1 prescaler value
  TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
}
```

Figure 3-6. Timer 1 peripheral initialization routine.

3.1.4.2 Timers algorithm

Timer 0 and Timer 1 ISR works similarly, with the only difference that Timer 0 periodicity is higher than Timer 1. Counters are decremented within Timers ISR, so multiplicity of base timers could be obtained (Figure 3-7).

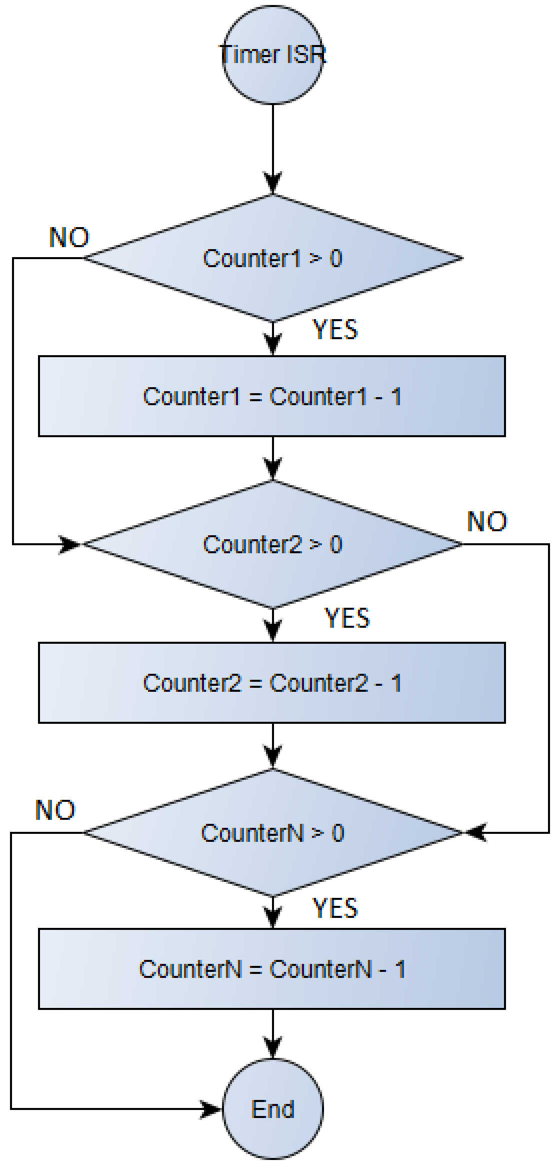


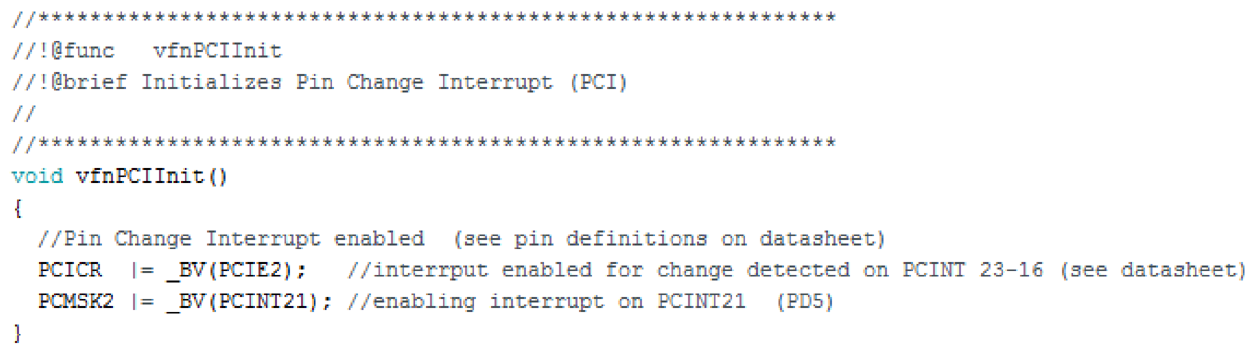
Figure 3-7. Timer 1 peripheral initialization routine.

3.1.5 PCI (Pin Change Interrupt)

This feature allows to detect any logic level change on the Atmega328p pin only if this function is enabled. There is the possibility to decrease the CPU load because interrupt feature works as a non-blocking mechanism, thus, CPU efficiency will be optimized. ATMEL, which is the manufacturer of the Atmega328p microcontroller, provides a schematic diagram to show which pins are PCI configurable.

3.1.5.1 PCI settings

PD5 (refer to Table 3-1) is designed to work as a logical input for the detection of pulse width duration of the ultrasonic sensor, so PCI features are need only on this pin. From Figure 3-8 corresponding interrupt for PD5 is PCINT21. PCI initialization routine is shown in Figure 3-9.



```
//**************************************************************
//!@func   vfnPCIInit
//!@brief Initializes Pin Change Interrupt (PCI)
//
//**************************************************************
void vfnPCIInit()
{
  //Pin Change Interrupt enabled  (see pin definitions on datasheet)
  PCICR  |= _BV(PCIE2);   //interrput enabled for change detected on PCINT 23-16 (see datasheet)
  PCMSK2 |= _BV(PCINT21); //enabling interrupt on PCINT21  (PD5)
}
```

Figure 3-8. Pin Change Interrupt peripheral initialization routine.

3.1.5.2 PCI algorithm

In order to avoid false readings due to electrical noise signals generated on logical level transitions (from high to low and vice-versa), a protection mechanism is implemented. This mechanism consists in waiting a short amount of time (milliseconds) before performing a second reading; if after two consecutive readings, the data remains unchanged, it could be considered as valid data. This protection mechanism is known as "debounce" (Figure 3-9).

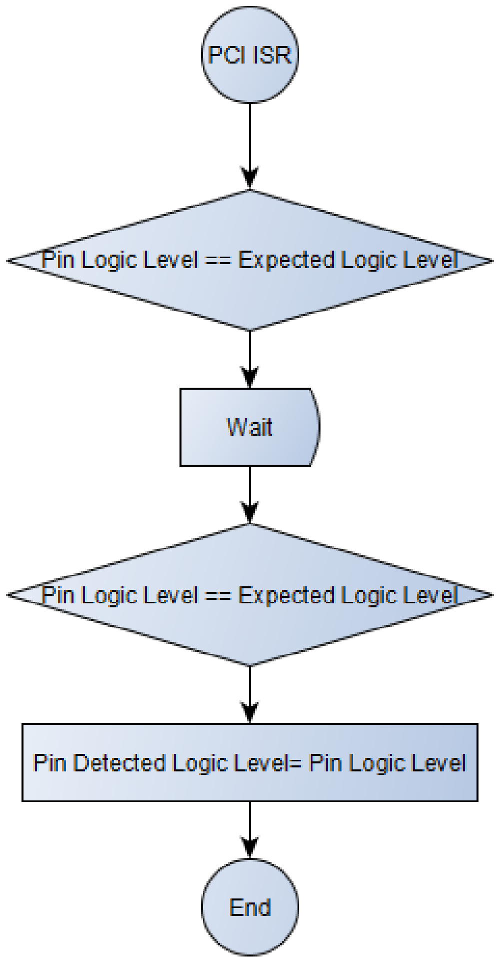
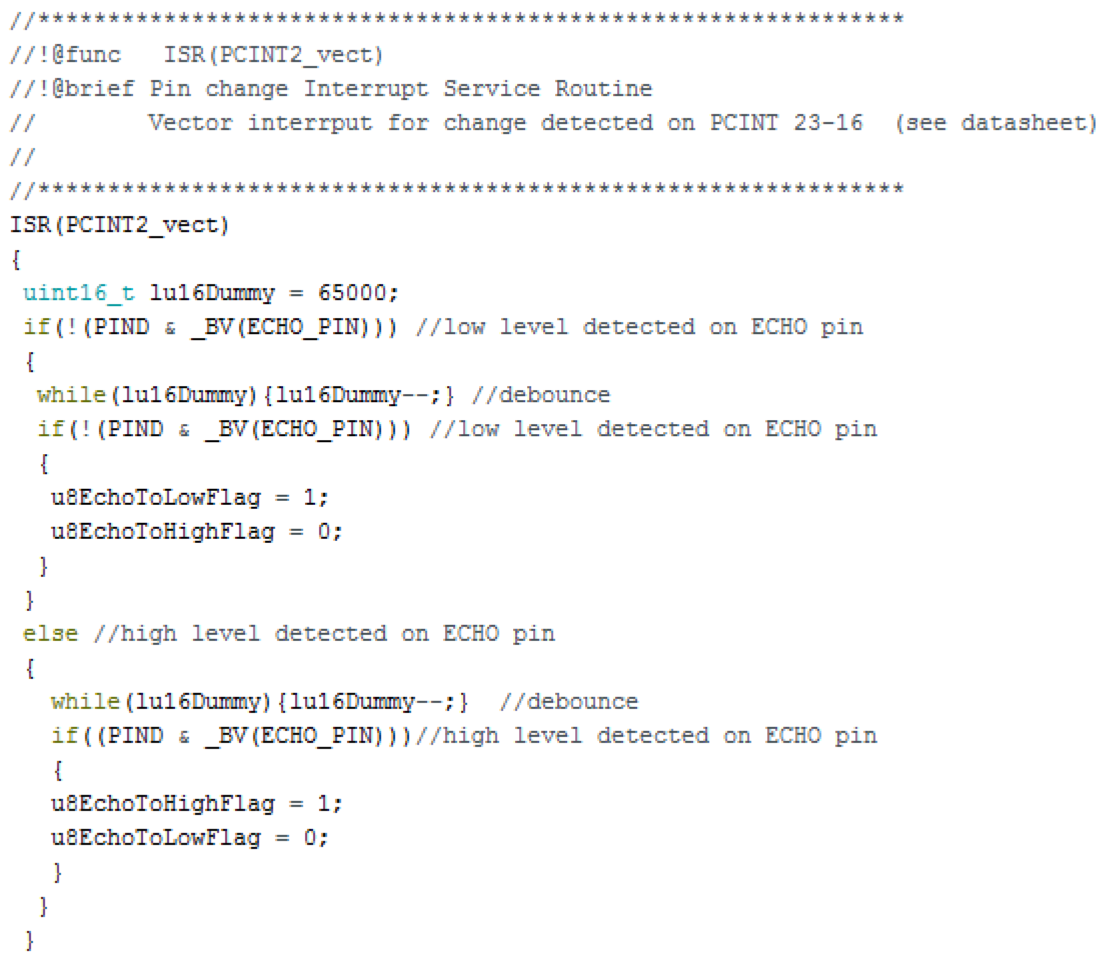

PCI ISR
Pin Logic Level == Expected Logic Level
Wait
Pin Logic Level == Expected Logic Level
Pin Detected Logic Level= Pin Logic Level
End

Figure 3-9. Pin Change Interrupt Service Routine (ISR) debounce mechanism.

PCINT2_vect contains PCINT23 to PCINT16, nevertheless, as only one PCI interrupt is implemented (PCINT21), the source of PCI is not evaluated (Figure 3-10).



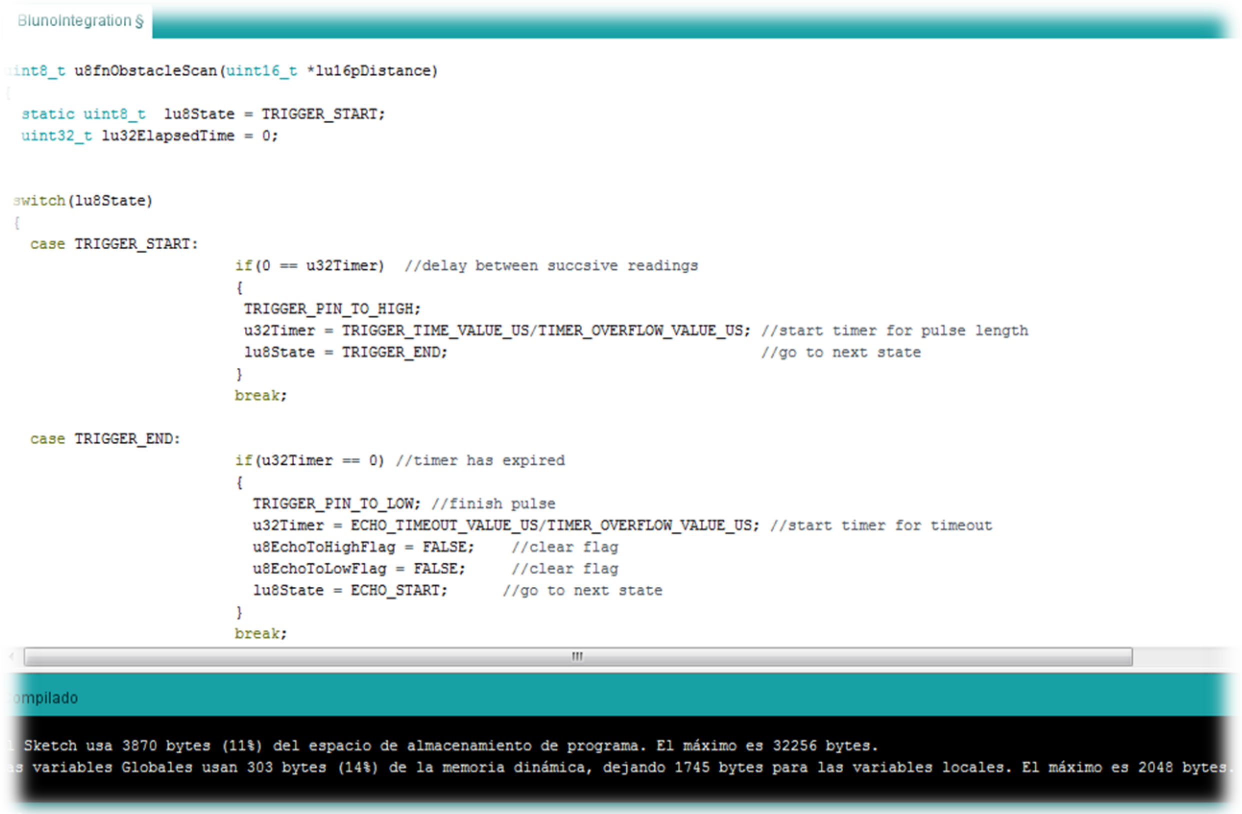
```
//**************************************************************
//!@func   ISR(PCINT2_vect)
//!@brief Pin change Interrupt Service Routine
//        Vector interrput for change detected on PCINT 23-16  (see datasheet)
//
//**************************************************************
ISR(PCINT2_vect)
{
 uint16_t lu16Dummy = 65000;
 if(!(PIND & _BV(ECHO_PIN))) //low level detected on ECHO pin
 {
  while(lu16Dummy){lu16Dummy--;} //debounce
  if(!(PIND & _BV(ECHO_PIN))) //low level detected on ECHO pin
  {
   u8EchoToLowFlag = 1;
   u8EchoToHighFlag = 0;
  }
 }
 else //high level detected on ECHO pin
 {
   while(lu16Dummy){lu16Dummy--;}  //debounce
   if((PIND & _BV(ECHO_PIN)))//high level detected on ECHO pin
   {
   u8EchoToHighFlag = 1;
   u8EchoToLowFlag = 0;
   }
  }
 }
```

Figure 3-10. Pin Change Interrupt Service Routine (ISR).

3.1.6 Firmware load and compiling processes

As mentioned earlier in this document, Arduino IDE (Integrated Development Environment) is used for code compiling and firmware load process, so the flash memory of the Bluno Beetle can be programmed (Figure 3-11). When this process is completed, DEBI is finally ready to work.

As shown in Figure 3-11, the compiler shows that 11% of program memory (flash memory) of the Atmega328p microcontroller has been used, and 14% of the data memory (volatile memory).


BlunoIntegration §

```
int8_t u8fnObstacleScan(uint16_t *lu16pDistance)
{
 static uint8_t  lu8State = TRIGGER_START;
 uint32_t lu32ElapsedTime = 0;


switch(lu8State)
{
  case TRIGGER_START:
                        if(0 == u32Timer)  //delay between succsive readings
                        {
                         TRIGGER_PIN_TO_HIGH;
                         u32Timer = TRIGGER_TIME_VALUE_US/TIMER_OVERFLOW_VALUE_US; //start timer for pulse length
                         lu8State = TRIGGER_END;                                   //go to next state
                        }
                        break;

  case TRIGGER_END:
                        if(u32Timer == 0) //timer has expired
                        {
                          TRIGGER_PIN_TO_LOW; //finish pulse
                          u32Timer = ECHO_TIMEOUT_VALUE_US/TIMER_OVERFLOW_VALUE_US; //start timer for timeout
                          u8EchoToHighFlag = FALSE;    //clear flag
                          u8EchoToLowFlag = FALSE;     //clear flag
                          lu8State = ECHO_START;      //go to next state
                        }
                        break;
```

ompilado

```
l Sketch usa 3870 bytes (11%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
s variables Globales usan 303 bytes (14%) de la memoria dinámica, dejando 1745 bytes para las variables locales. El máximo es 2048 bytes.
```

Figure 3-11. Arduino IDE compiling process.

3.2. Smartphone application

Directional lights (turn left and turn right) of DEBI are capable of being wirelessly activated through a smartphone running Android ™ OS (Operating System) and with Bluetooth 4.0 (also known as BLE [Bluetooth Low Energy]) feature, via a software application called DEBI app.

DEBI app is a smartphone application that allows the final user (cyclist) to activate the turn left indicator or turn right indicator of DEBI by tapping buttons on the screen. User Graphical Interface of DEBI app is shown in Figure 3-12.

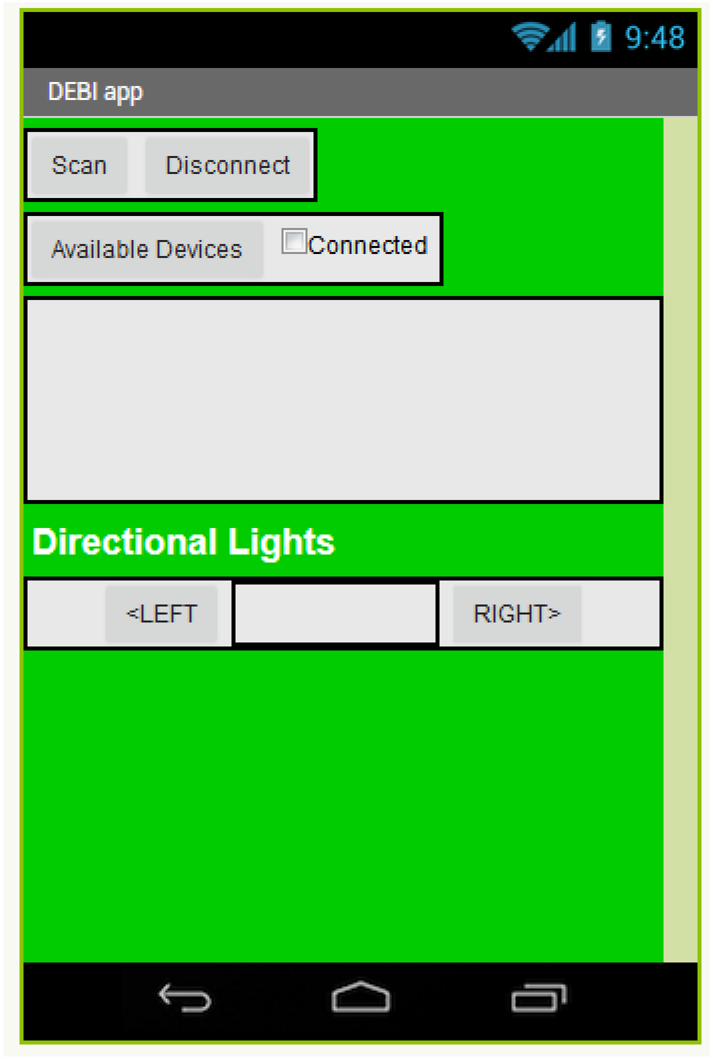
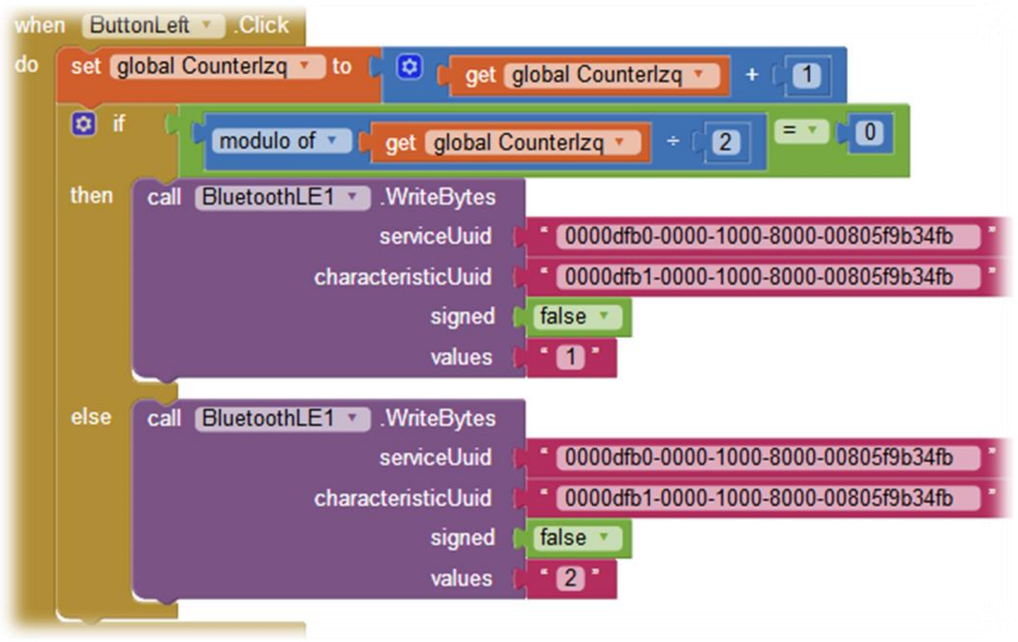


Figure 3-12. DEBI app interface.

For further versions of the DEBI application, voice commands are considered as replacement of “LEFT" and "RIGHT" buttons.

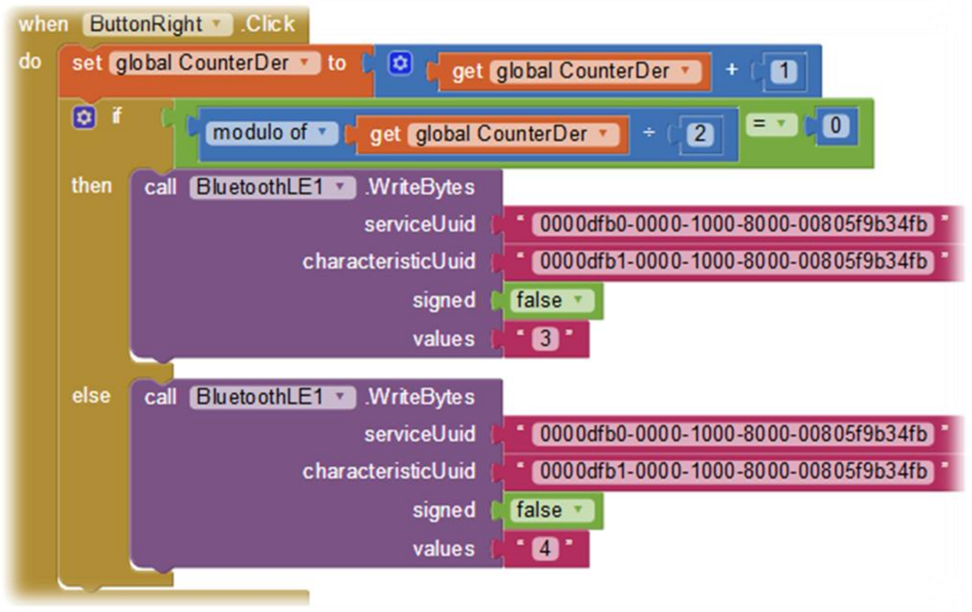
Form Figure 3-12, the "Scan" button will search for nearby BLE devices, then the available devices could be listed by tapping the “Available Devices" button; the “Connect" button will start a connection with the device selected on the available devices list and once a connection is created between the smartphone and BLE device, a checkbox will be marked; finally, "LEFT" or "RIGHT" buttons may be tapped in order to activate the corresponding DEBI’s directional lights.

Development of the DEBI app was made by using the MIT app inventor. MIT app inventor provides a visual programming language in form of puzzle blocks, so logical decisions could be performed based on user interface input controls, such as buttons and sliders (Figure 3-13 and Figure 3-14).



```
when ButtonLeft .Click
do  set global CounterIzq to  get global CounterIzq + 1
    if  modulo of get global CounterIzq ÷ 2 = 0
    then  call BluetoothLE1 .WriteBytes
                      serviceUuid  " 0000dfb0-0000-1000-8000-00805f9b34fb "
               characteristicUuid  " 0000dfb1-0000-1000-8000-00805f9b34fb "
                           signed  false
                           values  " 1 "
    else  call BluetoothLE1 .WriteBytes
                      serviceUuid  " 0000dfb0-0000-1000-8000-00805f9b34fb "
               characteristicUuid  " 0000dfb1-0000-1000-8000-00805f9b34fb "
                           signed  false
                           values  " 2 "
```

Figure 3-13. MIT app inventor. Blocks for programming left turn light.



```
when ButtonRight .Click
do  set global CounterDer to  get global CounterDer + 1
    if  modulo of get global CounterDer ÷ 2 = 0
    then  call BluetoothLE1 .WriteBytes
                      serviceUuid  " 0000dfb0-0000-1000-8000-00805f9b34fb "
               characteristicUuid  " 0000dfb1-0000-1000-8000-00805f9b34fb "
                           signed  false
                           values  " 3 "
    else  call BluetoothLE1 .WriteBytes
                      serviceUuid  " 0000dfb0-0000-1000-8000-00805f9b34fb "
               characteristicUuid  " 0000dfb1-0000-1000-8000-00805f9b34fb "
                           signed  false
                           values  " 4 "
```

Figure 3-14. MIT app inventor. Blocks for programming right turn light.

BLE uses GATT (Generic Attribute Profile), it defines the way that two Bluetooth Low Energy devices transfer data between them using concepts called Services and Characteristics. [17]

DFROBOT, which is the Bluno Beetle board manufacturer, provides services and characteristics values for starting up an example project [18]. These values are used as shown in Figure 3-13 and Figure 3-14.

Compiling application process could be done in two different ways, either by building .apk extension file and then transferring it to an Android OS device, or, by installing MIT app inventor app on the desired device to run the application and then, scanning a QR code provided after the application has been build.

3.1. Hardware design

In reference to hardware design, two printed circuit boards were developed for this purpose, using kicad v4.0.7 that is an open source electronics design suite [19]. This tool allowed to design the schematics and the creation of the printed circuit board (PCB). The complete schematic and PCB designs can be found in Appendix A. In Figure 3-15, a 3D view of DEBI's designs is shown.

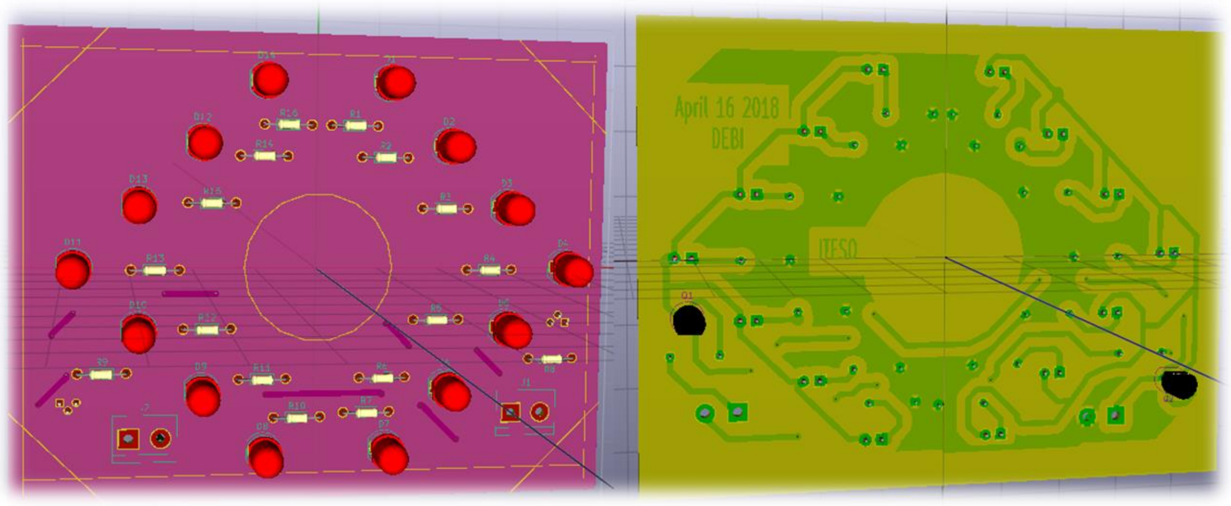


Figure 3-15. DEBI PCB 3D view from Kicad.

The boards needed for this project were created using a traditional method that consists in printing the circuits into a glossy paper and then using an iron at high temperature to transfer the toner from the paper to the copper board and the results are shown in Figure 3-16.

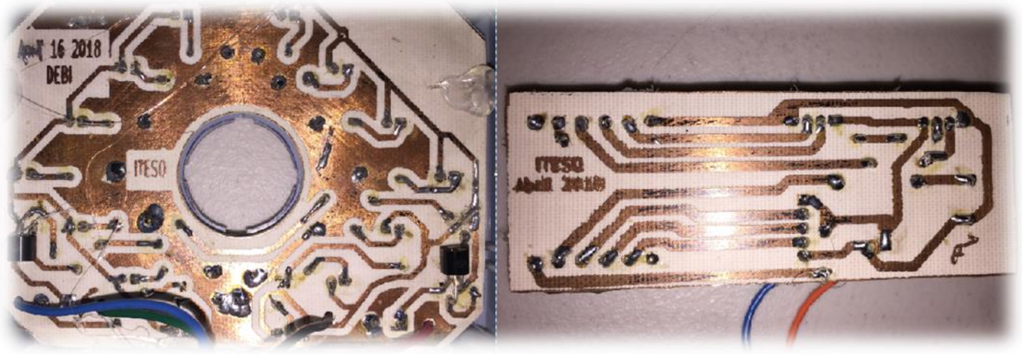


Figure 3-16. DEBI PCBs.

3.2. DEBI sensors

3.2.1 Accelerometer

As described in Table 2-3, this accelerometer has an operational voltage of 3.3V. However, it was used with the evaluation board that incorporates a 3.3V voltage regulator and can be supplied with 3V – 6V DC voltage (Figure 3-17).

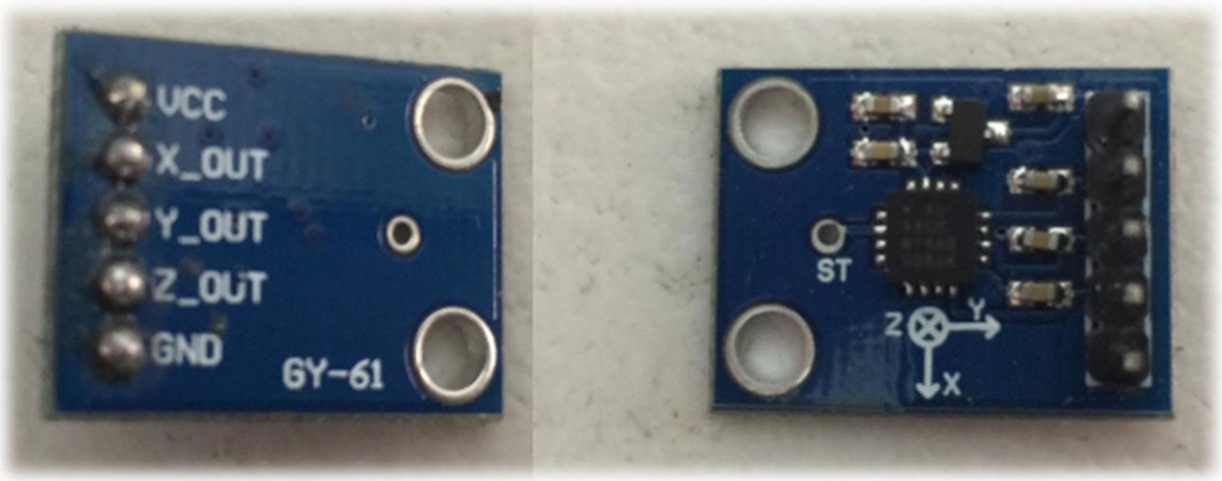


Figure 3-17. GY-61 sensor.

Since this is a 3-axis accelerometer, it can detect the changes in acceleration in 3 dimensions and the bandwidth is configurable using the Cy, Cx, and Cz capacitors located at Xout, Yout and Zout respectively. For this evaluation board, the value for those capacitors is 0.1uF and the bandwidth of each axis is 50Hz.

There is another pin in this evaluation board called ST, this pin controls the Self-Test feature that exerts an electrostatic force to the accelerometer bean when this pin is set to Vs resulting in the movement of the bean that allows the user to test if the accelerometer is functional. The typical change in output is: −1.08 g (corresponding to −325 mV) in the X-axis, +1.08 g (or +325 mV) on the Y-axis, and +1.83 g (or +550 mV) on the Z-axis. In normal use, this pin can be not connected or connected to ground [20].

Figure 3-18 refers to the algorithm used to determine when the user is stopping and the sop light is turned on.

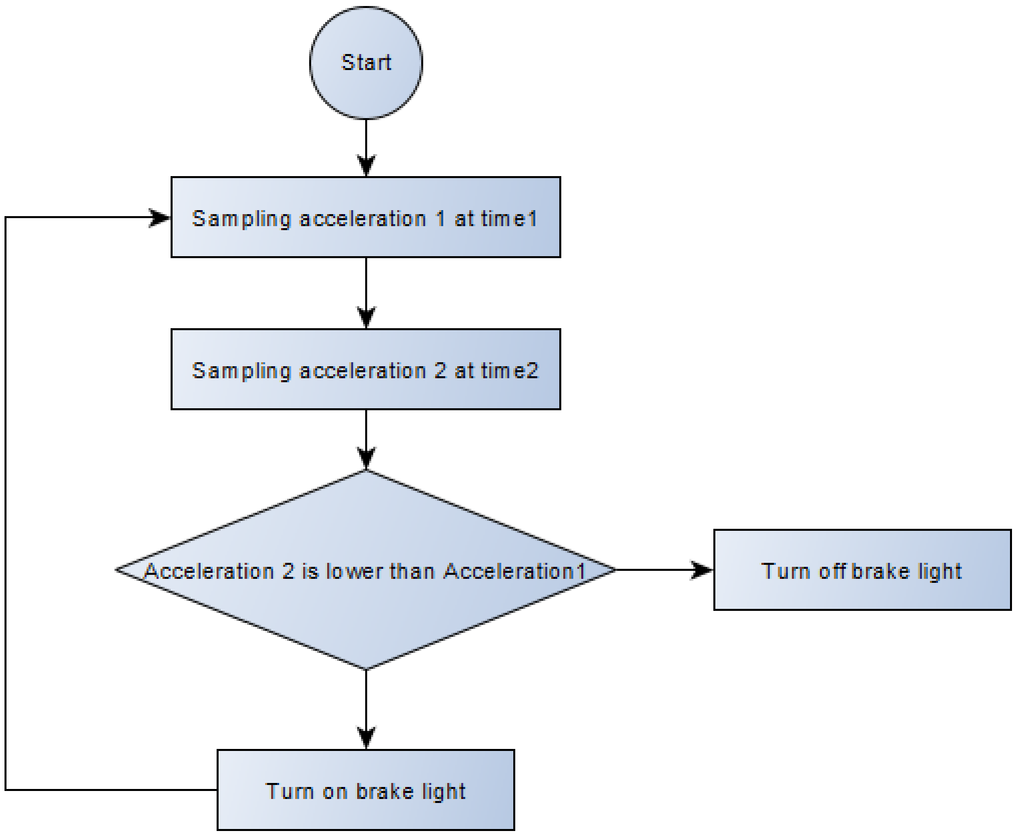

Start
Sampling acceleration 1 at time1
Sampling acceleration 2 at time2
Acceleration 2 is lower than Acceleration1
Turn off brake light
Turn on brake light

Figure 3-18. Brake light algorithm.

3.2.1 Proximity sensor

Figure 3-19 describes the procedure followed by the sensor module to obtain the distance measured between the detected object and sensor: first, a 10uS pulse is sent by the microcontroller and internally, eight 40KHz pulses are triggered by the module and once the last pulse is triggered, the ECHO pin switches from dominant to recessive state until it receives the echo back from an obstacle in front of the sensor or a timeout at 38mS, if no obstacle is found. Equations (3-8) and (3-9) show how to calculate the distance of an obstacle in centimeters and inches, respectively.

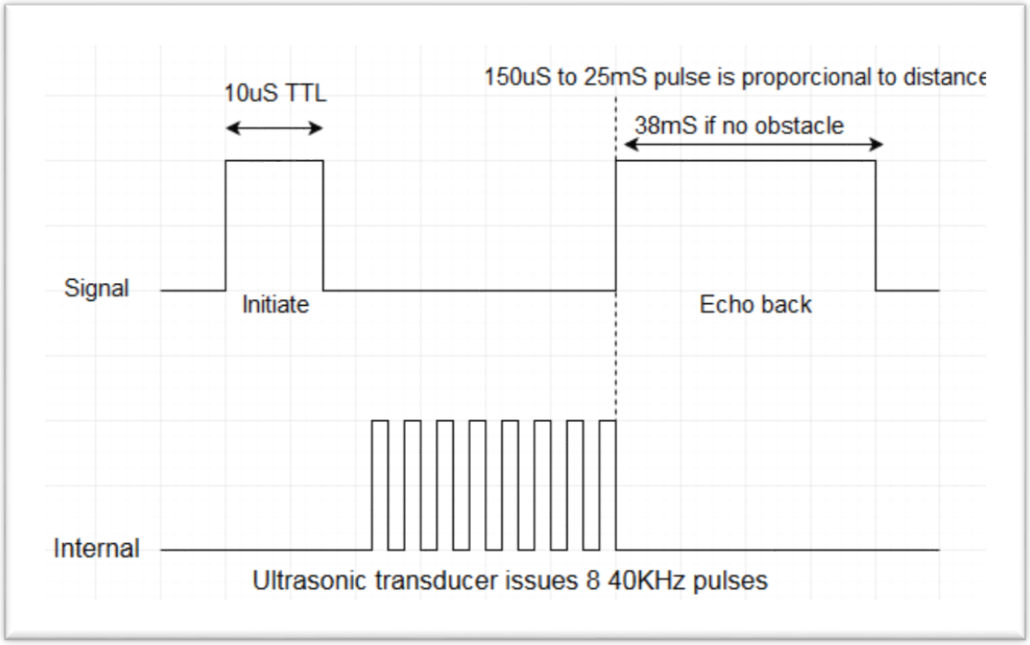

10uS TTL
150uS to 25mS pulse is proporcional to distance
38mS if no obstacle
Signal
Initiate
Echo back
Internal
Ultrasonic transducer issues 8 40KHz pulses

Figure 3-19. Ultrasonic timing diagram

$$distance\ (cm) = \frac{Pulse\ width\ \mu s}{58\ \ \mu s/cm} \tag{3-8}$$

$$distance\ (in) = \frac{Pulse\ width\ \mu s}{148\ \mu s/in} \tag{3-9}$$

Figure 3-20 shows the flow diagram with the process to identify vehicles in collision course.

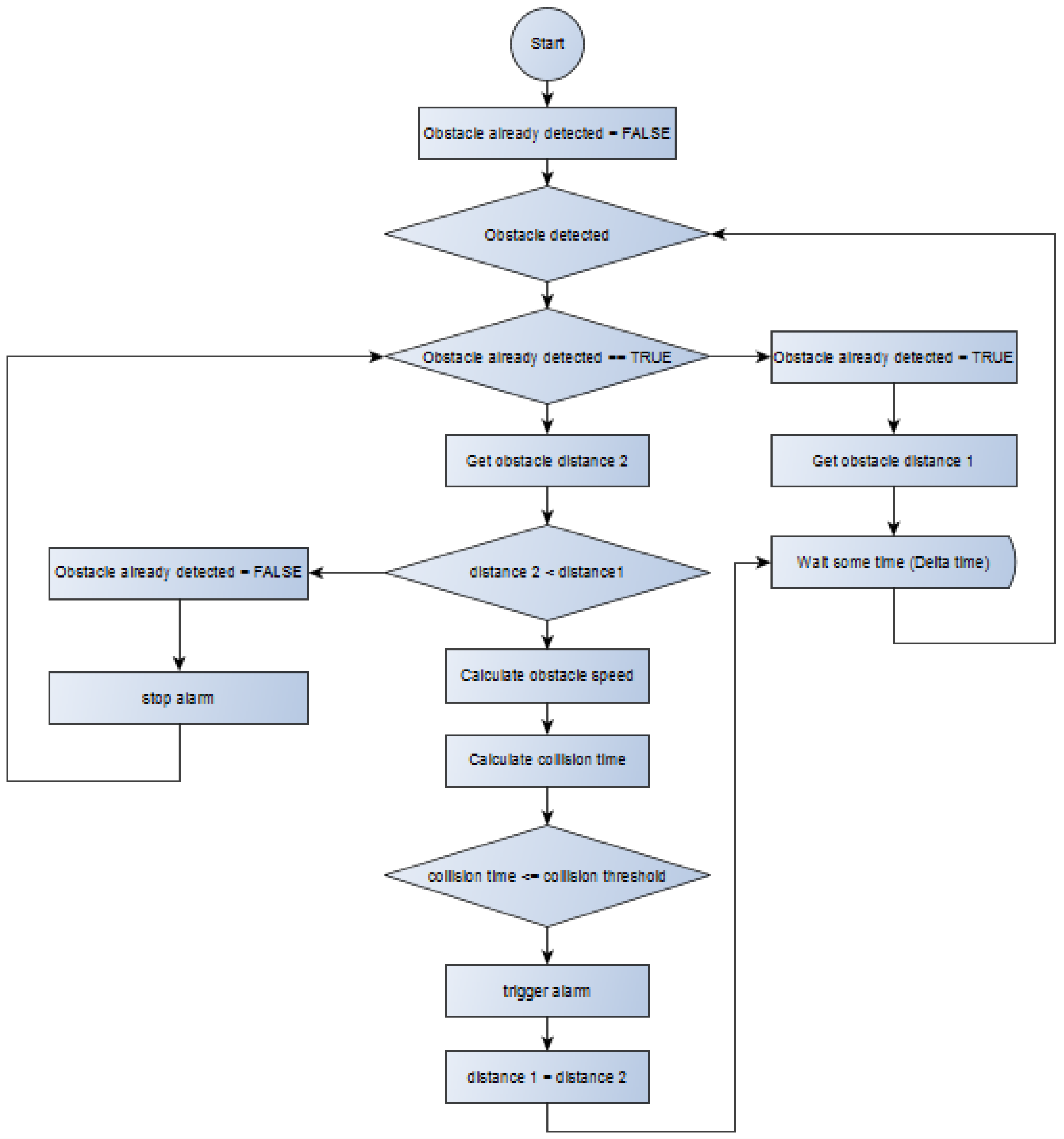

Start
Obstacle already detected = FALSE
Obstacle detected
Obstacle already detected == TRUE
Obstacle already detected = TRUE
Get obstacle distance 2
Get obstacle distance 1
Obstacle already detected = FALSE
distance 2 < distance1
Wait some time (Delta time)
stop alarm
Calculate obstacle speed
Calculate collision time
collision time <= collision threshold
trigger alarm
distance 1 = distance 2

Figure 3-20. Proximity sensor algorithm.

3.2.1 Directional lights

For directional lights the algorithm is shown in Figure 3-23:

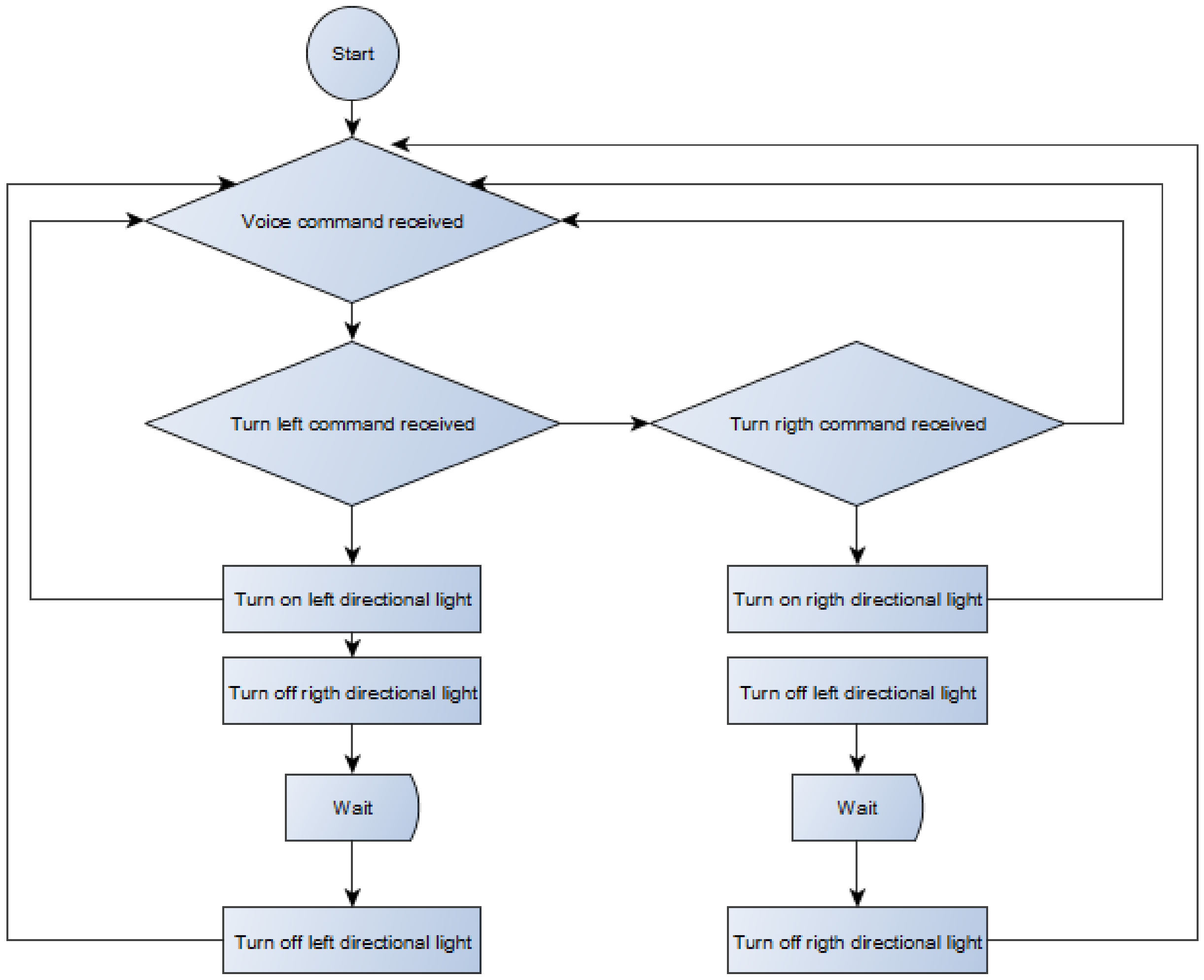

Start
Voice command received
Turn left command received
Turn rigth command received
Turn on left directional light
Turn on rigth directional light
Turn off rigth directional light
Turn off left directional light
Wait
Wait
Turn off left directional light
Turn off rigth directional light

Figure 3-21. Directional lights algorithm.

4. Results

In this section, the testing results performed in each of DEBI's features starting with the detection of approaching vehicles with the proximity sensor, are described. In this test, the objective was to prove that the device can detect when a vehicle is approaching and differentiate if it is in collision course by calculating the collision time. In reference to the inertial brake lights, it was demonstrated in what way the measurements from the accelerometers in the microcontroller are used to determine when the cyclist is stopping and finally, a description of directional lights behavior is provided.

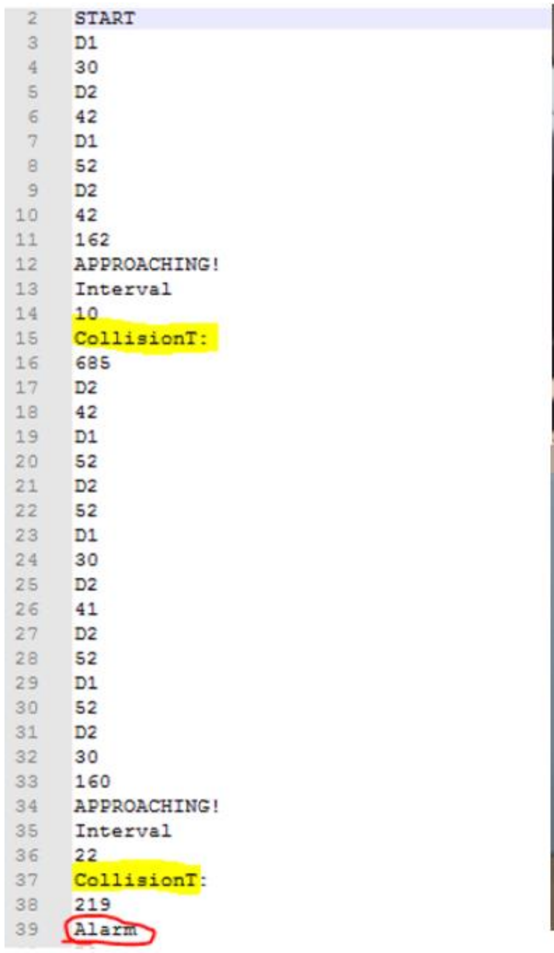
4.1. Proximity sensor testing

The collision time was calculated using the measurements from the proximity sensor to calculate the time of a possible collision. In Table 4-1, an example of this operation comparing the actual collision time with the collision time that is calculated by DEBI at a speed of 1m/s is shown. This device has an accuracy of approximately 90%.

Table 4-1. Proximity sensor tests (1m distance, approaching at a speed of 1m/s).

Calculated collision time (ms)	**Measured collision time (ms)**	**Accuracy of calculation (%)**
1100	1025	93.18
912	790	86.62
1100	1060	96.36
897	820	91.42
1440	1117	77.57
817	720	88.13
1200	1410	65.46
770	830	92.77
1140	1100	96.49
816	990	82.42

Figure 4-1 is an extract from the data logged by DEBI; it was obtained using the serial port while DEBI was connected to the PC. Distances detected between DEBI and the obstacle (D1 and D2) are shown. When the device detects that D2 is less in distance than D1, it assumes that the vehicle is approaching and calculates the collision time. If the collision time is higher than 500 ms, it is not considered as a possible collision, otherwise an alarm is raised to alert the cyclist.



```
START
D1
30
D2
42
D1
52
D2
42
162
APPROACHING!
Interval
10
CollisionT:
685
D2
42
D1
52
D2
52
D1
30
D2
41
D2
52
D1
52
D2
30
160
APPROACHING!
Interval
22
CollisionT:
219
Alarm
```

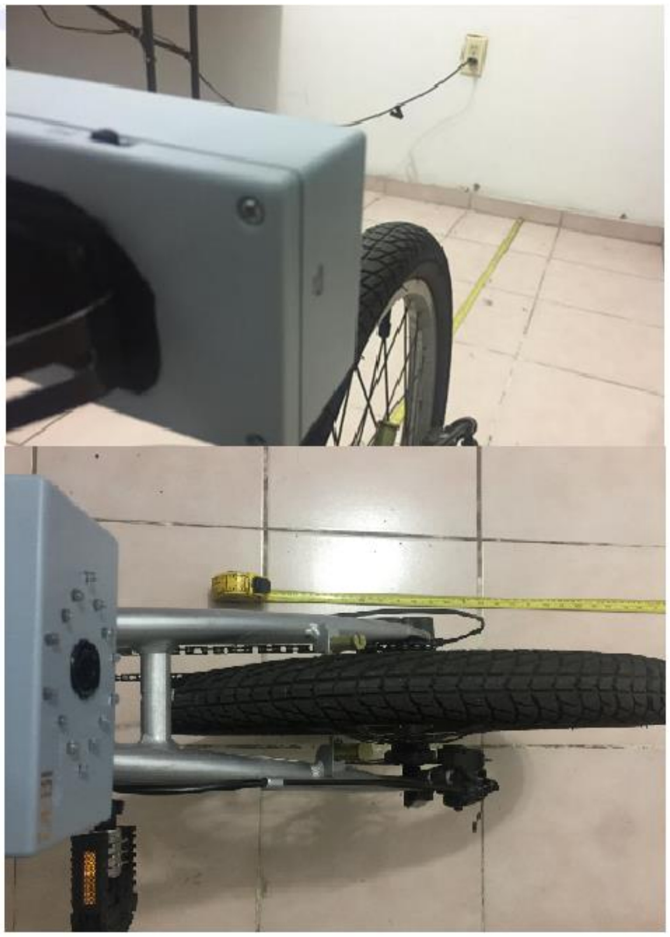


Figure 4-1. Proximity sensor readings from serial port using a personal computer (PC)

4.2. Brake lights testing

As described in the last chapter, DEBI has a built-in accelerometer and it is used to detect when the cyclist is stopping and thus, turns on the lights, to do this, the microcontroller is continually sensing the acceleration of the Z axis and when a change in acceleration is detected higher than N samples in M time, the alarm is raised.

It is important to mention that N, M, and configurable values for this test. In order to perform calibration, they were set empirically to: N = 30 ADC samples and M = 100 ms.

Additionally, a voltmeter was used to determine values for: 0g, 1g, and -1g and these can be extrapolated to obtain the values for the complete range of this accelerometer in the axis Z (Table 4-2).

Table 4-2. Table of equivalences for the Accelerometer's Z axis.

g forces	-3	-2	-1	0	1	2	3
Volts	0.0603	0.957	1.311	1.665	2.019	2.373	2.727
ADC samples	144	221	298	375	452	529	606

From Table 4-2, the scale factor for the application, that is 354V/g, which according to the datasheet must be in the range of (360mV/g - 195mV/g), depending on the power supply used (Vs from 2V to 3.6V) [21].

Then, we have that 30 ADC samples are equal to .39g and if we consider a value of g= $9.81 {}^{m}/_{s^2}$.(Equation (4-1)).

$$30\ ADC\ samples = 3.82\ {}^{m}/_{s^2} \tag{4-1}$$

Now, the increment on the speed is needed to indicate that the cyclist is braking, and this is calculated according to the Equation (4-2):

$$\Delta V = at \tag{4-2}$$

Where $\Delta V$is the change on the velocity, a is the acceleration and t is the time, it may be assumed that the acceleration was constant during this period of time. Then, replacing these variables with the obtained values, the Equation (4-3) can be obtained:

$$\Delta V = \left(3.82\ {}^{m}/_{s^2}\right) * .100s\ \approx\ .38\,{}^{m}/_{s}\ \approx \mathbf{1.37}\,{}^{km}/_{h} \qquad 4\text{-}310)$$

The above result means that the current configuration of DEBI’s stop light can detect when the cyclist decreases the speed in approximately 1km/h and thus, turns on the stop light. (Figure 4-2).

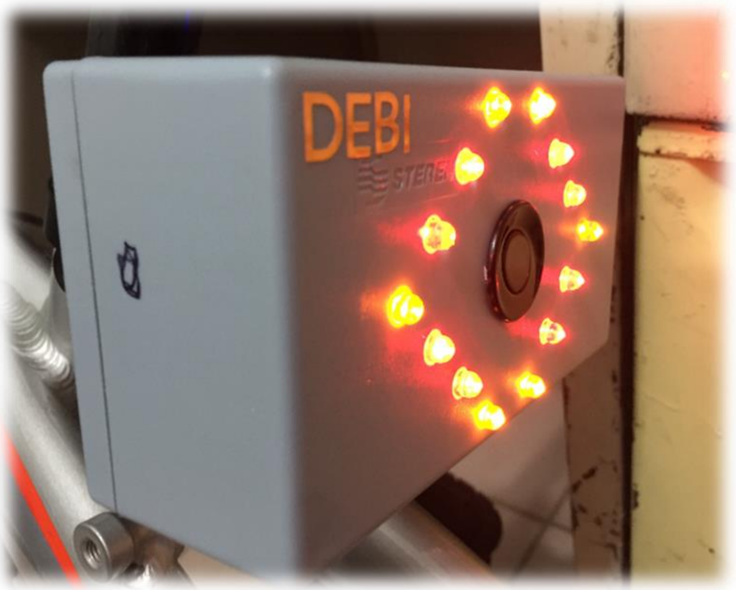


Figure 4-2. DEBI's brake light.

4.3. Turn lights

Another important security feature of DEBI is the capability to indicate the cyclist´s intention of turning left of right. This feature is very relevant since this allows other drivers become aware about the imminent cyclist course and take actions as needed in order to avoid any accident. Testing for DEBI’s turn lights was performed as follows:

1- Turn on DEBI.

2- Open DEBI’s app from an Android Smartphone.

3- Tap on the SCAN button to begin scanning for the Bluetooth devices available in the range

4- Tap on AVAILABLE DEVICES button and select the appropriate, in this case “Bluno Beetle”.

5- Once the device is connected, the checkbox in DEBI app will be marked, this means that DEBI has been successfully paired.

6- Now both buttons, LEFT and RIGHT are available to control the DEBI lights as shown in figure 4-3.

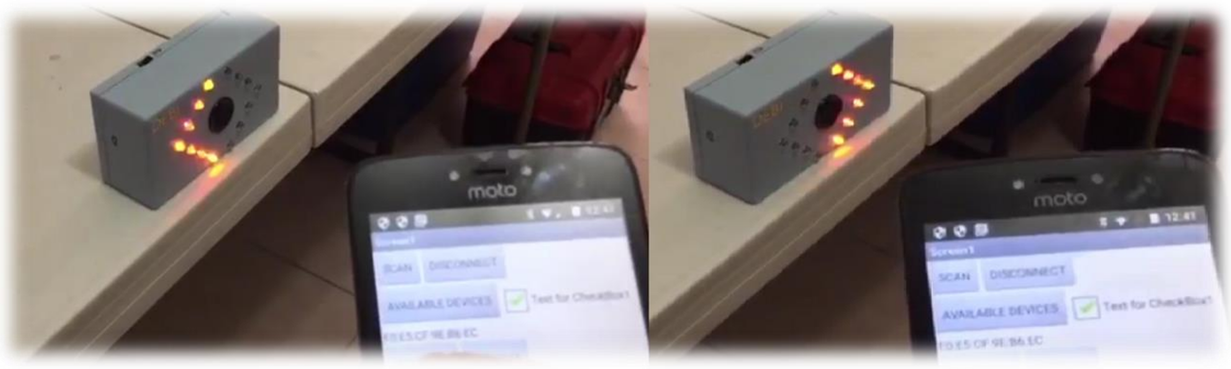


Figure 4-3. DEBI Turn lights

It is important to mention that the maximum distance for Bluetooth communication is 50m according to the manufacturer [22], but Bluetooth range is not crucial for this application since the cyclist needs to keep the smartphone near the bicycle in order to control the turn lights, then a maximum distance about 1m will be enough.

5. Conclusions

Based on measurements and results obtained in the previous chapter, it was demonstrated that collision time of an incoming object could be calculated with an effectiveness of almost 90% by using low cost (below 1000 pesos) sensors, however two issues were detected: false readings and short range of the proximity sensor. The first issue can be handled via software, this implies investing more CPU processing time for acquiring more samples. Nevertheless, if this solution is implemented, time for making speed calculations of incoming objects will be increased. Then, the maximum speed of incoming vehicles at which DEBI can react will decrease. For example, consider a distance sensor with a maximum range of 5 meters and an alarm that will trigger if the calculated collision time for a vehicle approaching DEBI is lower or equal than 1 second, and the time needed for making calculations is 100 ms, this means DEBI will has not enough time to make the necessary calculations and thus, trigger an opportune alarm if a vehicle approaches at a speed that represents a collision time higher than 1.1 seconds. For this example that means a speed of 16 km/h approximately. Now, suppose that the number of samples that DEBI takes is increased at triple with the purpose of making accurately calculations, then the time for making calculations also becomes triplicated resulting in 300 ms, consequently, the maximum speed at which DEBI can react has decreased to lower than 14km/hr.

On the other hand, the maximum range of detection is a matter of hardware so there is nothing to do in this matter via software. On the same line, it is important to say that false detection and maximum range of detection could be improved with a robust proximity sensor that allows to decrease samples needed and increase the detection range so the speed of vehicles at which DEBI can react could be increased or the collision time threshold could be increased, nevertheless, a robust sensor implies higher cost.

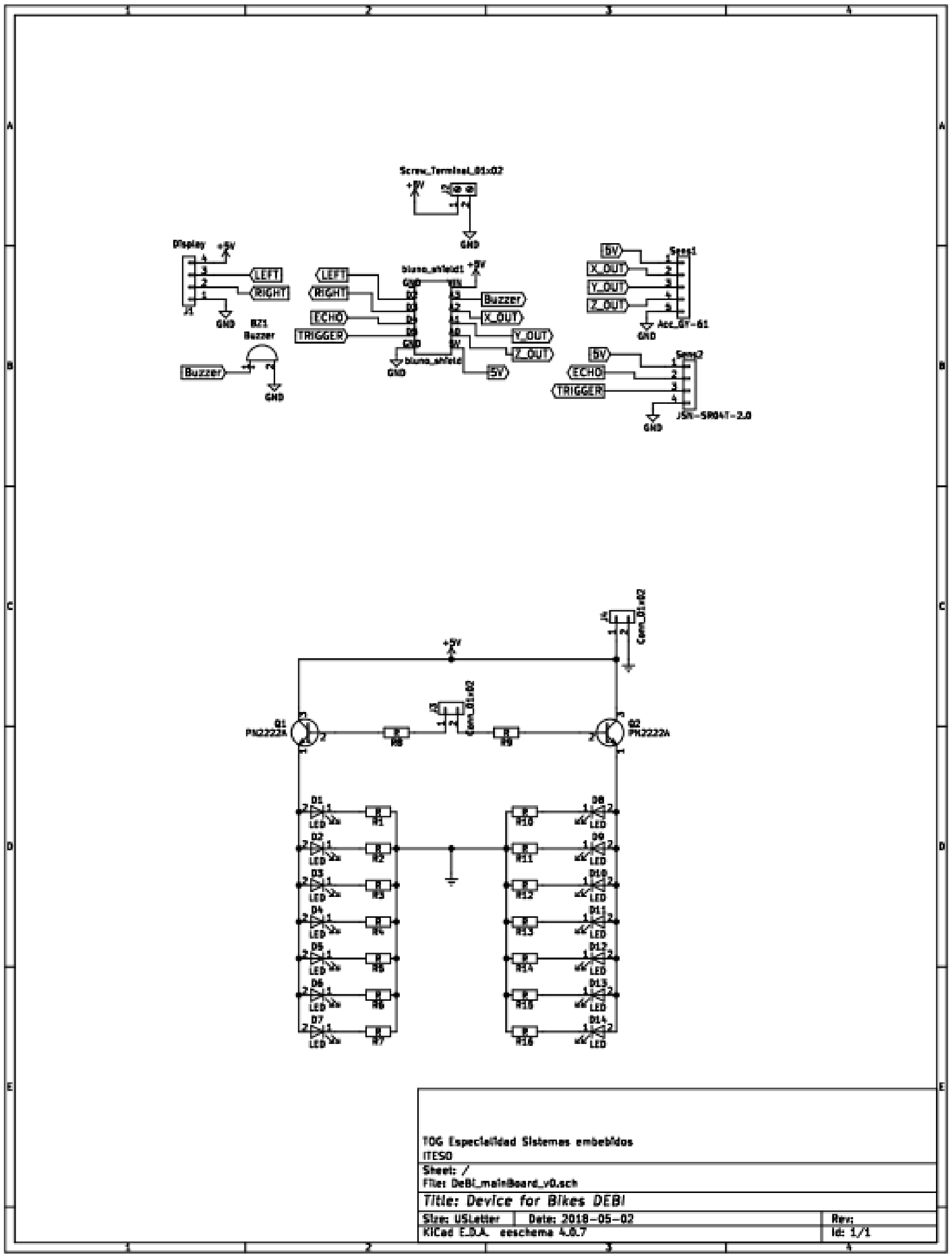
In reference to the stop lights, the results for the accelerometer show that it is reliable (more than 90% accuracy) and this could also have more applications in future versions of this project, like identifying if the cyclist had an accident and notify to the emergency services using

Bluetooth communication with a smartphone. DEBI can be very useful for a cyclist and it can be used as a remote to control turn lights. Finally, the DEBI prototype was a low-cost system (less than 1000 pesos up to date), however, it is not reliable as a final product. In order to make it a reliable product (more than 90 % accuracy in all sensors), a more robust sensor has to be considered and validated, and that also implies DEBI's cost will proportionally be increased.

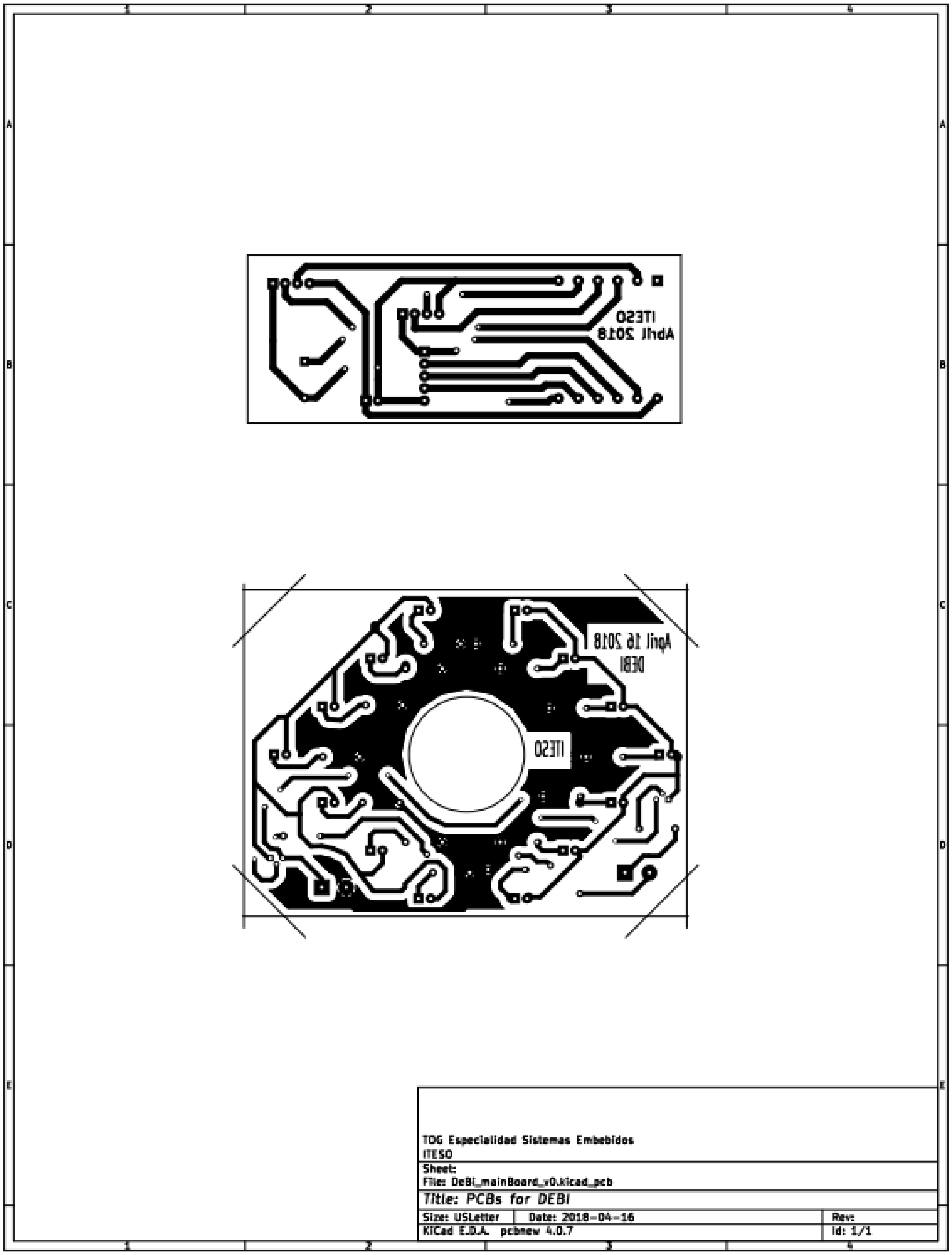
Appendix

Appendix A

DEBI electrical diagram



DEBI Printed Circuit board


TDG Especialidad Sistemas Embebidos
ITESO
Sheet:
File: DeBi_mainBoard_v0.kicad_pcb
Title: PCBs for DEBI
Size: USLetter
Date: 2018-04-16
Rev:
KiCad E.D.A. pcbnew 4.0.7
Id: 1/1

References

[1] “MIBICI | How it works?” [Online]. Available: https://www.mibici.net. [Accessed: 17-Jun-2018].

[2] “Inicio | Manual del Ciclista.” [Online]. Available: https://www.jalisco.gob.mx/ciclista/. [Accessed: 12-Jun-2018].

[3] Latin American Housing Network, “Dinámica demográfica del área metropolitana de Guadalajara,” p. 4.

[4] John and EJ Craig, “Lucid Brake,” *LucidBrake 2.0-Motion-Sensing, Press-On Bicycle BRAKE Light*. .

[5] “Blinkers,” *Blinkers*. [Online]. Available: https://www.blinkers.bike/. [Accessed: 17-Jun-2018].

[6] Garmin and G. L. or its subsidiaries, “Varia Rearview Radar,” *Garmin*. [Online]. Available: https://buy.garmin.com/en-US/US/p/518151. [Accessed: 17-Jun-2018].

[7] “michelin’s ‘bikesphere’ protects cyclists from cars inside a ring of red light.” [Online]. Available: https://www.designboom.com/technology/michelin-trendy-drivers-bikesphere-06-09-2017/. [Accessed: 17-Jun-2018].

[8] “Mi Bici costará un peso por día | Gobierno del Estado de Jalisco.” [Online]. Available: https://www.jalisco.gob.mx/es/prensa/noticias/17017. [Accessed: 17-Jun-2018].

[9] “Plan estatal de desarrollo Jalisco.” [Online]. Available: http://seplan.app.jalisco.gob.mx/biblioteca/panel/index. [Accessed: 17-Jun-2018].

[10] R. Toulson and T. Wilmshurst, *Fast and Effective Embedded Systems Design : Applying the ARM Mbed*. Oxford: Newnes, 2012.

[11] Atmel, “ATmega328P - 8-bit AVR Microcontrollers - Microcontrollers and Processors.” [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega328P. [Accessed: 21-Jun-2018].

[12] “Arduino - Introduction.” [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed: 21-Jun-2018].

[13] “Bluno Beetle SKU:DFR0339 - DFRobot Electronic Product Wiki and Tutorial: Arduino and Robot Wiki-DFRobot.com.” [Online]. Available: https://www.dfrobot.com/wiki/index.php/Bluno_Beetle_SKU:DFR0339. [Accessed: 21-Jun-2018].

[14] R. Estrada, “ADXL335 GY61 Acelerómetro,” *HeTPro*. [Online]. Available: https://hetpro-store.com/adxl335-gy61/. [Accessed: 21-Jun-2018].

[15] “Proximity Sensors Compared: Inductive, Capacitive, Photoelectric, and Ultrasonic,” *Machine Design*, 01-Sep-2001. [Online]. Available:

http://www.machinedesign.com/sensors/proximity-sensors-compared-inductive-capacitive-photoelectric-and-ultrasonic. [Accessed: 26-Jun-2018].

[16] “About Us | Explore MIT App Inventor.” [Online]. Available: http://appinventor.mit.edu/explore/about-us.html. [Accessed: 11-Jul-2018].

[17] “GATT | Introduction to Bluetooth Low Energy | Adafruit Learning System.” [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt. [Accessed: 11-Jul-2018].

[18] “Bluetooth APP Control - DFRobot Electronic Product Wiki and Tutorial: Arduino and Robot Wiki-DFRobot.com.” [Online]. Available: https://www.dfrobot.com/wiki/index.php/Bluetooth_APP_Control. [Accessed: 11-Jul-2018].

[19] “KiCad EDA.” [Online]. Available: http://kicad-pcb.org/. [Accessed: 11-Jul-2018].

[20] “ADXL335.pdf.” .

[21] Analog Devices, “ADXL335.pdf.” [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf. [Accessed: 21-Jun-2018].

[22] “Beetle BLE - The smallest Arduino Bluetooth 4.0 (BLE) - DFRobot.” [Online]. Available: https://www.dfrobot.com/product-1259.html. [Accessed: 11-Jul-2018].