Prototipado rápido para monitoreo IoT a traves de Bluetooth y aplicaciones móviles

José A. Hernández-Benitez[1,2], Jose R. Atoche-Enseñat[1], Jorge C. Canto-Esquivel[1], Ricardo J. Peón-Escalante[2,*], Jaime F. Avilés-Viñas[2], Renán G. Quijano-Cetina[2], Orlando Palma-Marrufo[2], Alejandro A. Castillo-Atoche[2]

[1] *Departamento de ingeniería eléctrica y electrónica Tecnológico Nacional de México/ITMérida*
[2] *Facultad de Ingeniería, Universidad Autónoma de Yucatán, A.P. 150, Cordemex, Mérida, Yucatán, México.*

Fecha de recepción: 9 de noviembre de 2022 — Fecha de aceptación: 11 de diciembre de 2022

Resumen

La necesidad de realizar mediciones en procesos industriales desde una aplicación remota ha crecido en los últimos años con el desarrollo del Internet de las cosas (IoT). En este sentido, es de gran interés conocer una metodología rápida para desarrollar sistemas embebidos que adquieran señales provenientes de sensores y otros dispositivos, y que los datos puedan ser monitoreados en tiempo real, de manera interactiva, y desde dispositivos personales como computadoras portátiles o teléfonos móviles. En este trabajo se propone una metodología para el prototipado rápido de una red de sensores, y se puedan monitorear de manera remota los datos con un microcontrolador. El protocolo Bluetooth es utilizado en la transmisión de datos, y la herramienta MIT App Inventor se emplea para el desarrollo de una aplicación móvil.

Palabras Clave: Internet de la cosas, aplicación movil, redes inalámbricas.

Fast prototyping for IoT monitoring through Bluetooth and mobile applications

Abstract

The need for measurements in industrial processes from a wireless sensor network has grown in the last years with the development of the Internet of things. Then, the design implementation with a rapid prototyping methodology is vital for the embedded system development with data acquisition from sensors and other devices. These measurements must be interactive for the users through handheld devices such as personal computers or mobile appliances. In this work, a fast prototyping methodology for the implementation of a wireless sensor network is proposed. Experiment results demonstrate a quick integration of a low-power microcontroller, a Bluetooth transceiver for data transmission, and the development of a mobile application with the MIT App Inventor tool.

Keywords: Fast prototyping, internet of things, wireless networks.

1 Introduction

The Internet of Things (IoT) is one of the main concepts that support the fourth industrial revolution or industry 4.0. IoT is composed of hardware and software modules that allow an object or thing to be connected to the cloud. In this sense, the development of electronic embedded systems has allowed measuring almost any kind of physical phenomena using transducers and analog to digital converters. After that, a microcontroller typically processes these signals to be transmitted through wireless devices. In the cloud, this information needs to be interactive for the user, representing mobile phones as a good fit for data control and management. However, designing a wireless sensor network with the incorporation of electronic systems and user interfaces has a lot of

*rpeon@correo.uady.mx

Nota: Este artículo de investigación es parte de Ingeniería–Revista Académica de la Facultad de Ingeniería, Universidad Autónoma de Yucatán, Vol. 26, No. 3, 2022, ISSN: 2448-8364

alternatives. Therefore, designers need to know how to build prototypes rapidly. Monitoring applications with the IoT paradigm have been reported in the literature during the last few years. For example, [Mohammadi and Rashidzadeh, 2021] propose a battery system monitoring for electric vehicles to improve the efficiency of the state of charge. In [Kodali et al., 2020], a system for measuring temperature, humidity, and food quality from grain storage was developed to monitor the food state and to avoid waste and losses. [Abdullah et al., 2022] develops a system for blood saturation, heart rate, pulse rate, and body temperature to monitor and help the medical staff with data measurements of COVID-19 patients.

In the hardware part, [Meruje, 2018] gives benchmarks with Raspberry and Arduino boards with Wi-Fi wireless transmission, but the user interfaces part is missing. Arduino Board has grown in popularity, and the technical forums provide a lot of suggestions in case of problems. After that, the wireless protocol is selected, and the application interface is developed to complete the approach for an IoT monitoring system.

For the network, Bluetooth has emerged as a short-range standard for data transmission due to good bandwidth and low energy consumption capabilities. The evolution to the 5.0 version has pushed future IoT scenarios with Bluetooth [Collotta et al., 2018]. The accomplishment of long-range coverage through short-range classical Bluetooth links is tackled in [Bansal et al., 2021].

Mobile devices have been used as interactive interfaces for remote sensing analysis. MIT App Inventor has grown in the IoT community [St. Georgiev, 2019] due to their free online development platform and mainly for Java block programming under Android operative systems. A good review for App programming is proposed by [Mir and Llueca, 2020]. In [Adiono et al., 2019], MINDS-app V1 was developed for data controlling and monitoring within the STM32 microcontroller and Xbee module. However, this wireless technology is less used, and energy-hungry devices are employed. A web platform was also presented in [Munasinghe et al., 2019]. The friendly interface has personalized services using IoT-enabled smart things.

In this work, the motivation consists of an IoT-based rapid prototyping monitoring system integrated with a microcontroller and a Bluetooth transceiver for sending and receiving data from a mobile phone application developed with the MIT App Inventor framework. This concept is illustrated in Figure 1.

The rest of the paper is organized as follows: Section 2 shows a brief introduction to Bluetooth protocol. The interconnection and programming of the HC-05 Bluetooth module and their interconnection with the microcontroller are illustrated. In Section 3, the development of two applications, including the mobile user interface with the MIT App Inventor is presented. Finally, the concluding remarks are given.

2 Methodology

2.1 Bluetooth overview.

Bluetooth is a standard for short-range, low-power, and low-cost wireless communication that uses radio technology. This Wireless Personal Area Network (WPAN) technology is now an IEEE standard under the 802.15 WPANs denomination. Today, this technology is embedded in numerous types of personal devices.

A Bluetooth Wireless Personal Area Network (BT-WPAN) consists of piconets. Each piconet is a cluster of up to eight Bluetooth devices. One device is designated as the master, and the others are slaves. Moreover, two piconets can be connected through a gateway or bridge Bluetooth device to form a scatter net. The specification divides the Bluetooth protocol stack into three logical groups. They are the Transport Protocol group, the Middleware Protocol group, and the Application group, as shown in Figure 2.

The Transport protocols allow Bluetooth devices to locate each other and manage physical and logical links with higher-layer protocols and applications. The Radio, Baseband, Link Manager, Logical Link Control and Adaptation (L2CAP) layers, and the Host Controller Interface (HCI) are included in the Transport Protocol group.

The Middleware Protocol group includes third-party and industry-standard protocols and Bluetooth SIG-developed protocols. The serial port emulator (RFCOMM) is an example of the last one.

The Host Controller Interface (HCI) layer defines a standard interface for upper-level applications to access the lower layers of the stack. Its purpose is to enable interoperability among devices.

A Bluetooth transceiver is a frequency hopping spread-spectrum (FHSS) device that uses the unlicensed (worldwide) 2.4 GHz ISM (Industrial, Scientific, Medical) frequency band. The nominal bandwidth for each channel is 1MHz. It has a range of approximately 10 meters. Bluetooth devices each have a unique Global ID used to create a hopping pattern. The master radio shares its Global ID and clock offset with each slave in its piconet, providing the offset into the hopping pattern. Bluetooth device can be configured in one of the following states: standby, inquiry, page, connected, transmit, hold, park or sniff. See Figure 3 for details.

Standby mode is configured when powered on but has not yet joined a piconet. It enters an Inquiry state when sending out requests to find other devices to which it might connect. A master of an existing piconet may

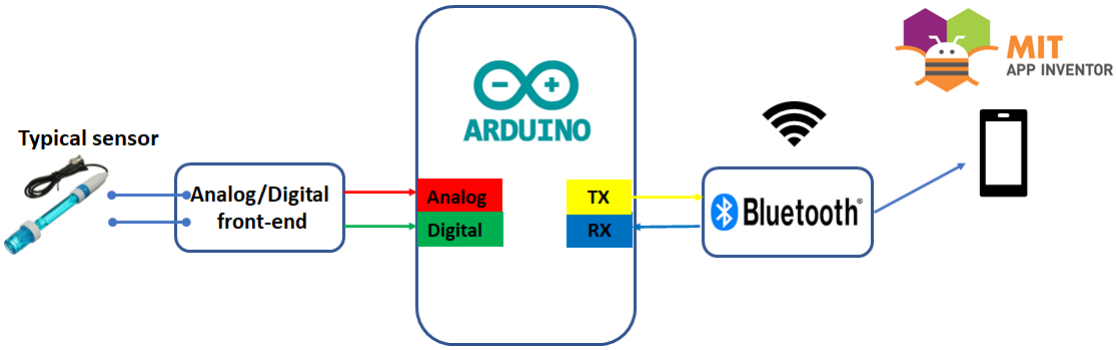

Figure 1: IoT monitoring system.

also be in a Page state, sending out messages looking for devices that it can invite to join its piconet. When successful communication is made between the master and the new device, the new device assumes the slave role, enters the Connected state, and receives an active address. While connected, the slave can transmit data when the master polls it. During the transmission of its data, the slave is in a Transmit state. At the end of its transmission, it returns to the Connected state.

The Sniff state works in a low-power consumption state when the slave "sleeps" for a pre-determined number of time slots. It wakes up at its appointed time slot for data transmission. It then returns to the inactive state until its next designated Sniff time slot arrives. The Hold state is another low-power state in which the slave is not active for a predetermined amount of time. However, there is no data transfer within the Hold state.

When a slave device has no data to send or receive, the master may instruct the slave to enter the Park state. When it enters a Park state, the slave relinquishes its active address in the piconet. The address will then be given to another slave that the master is reactivating from Park state.

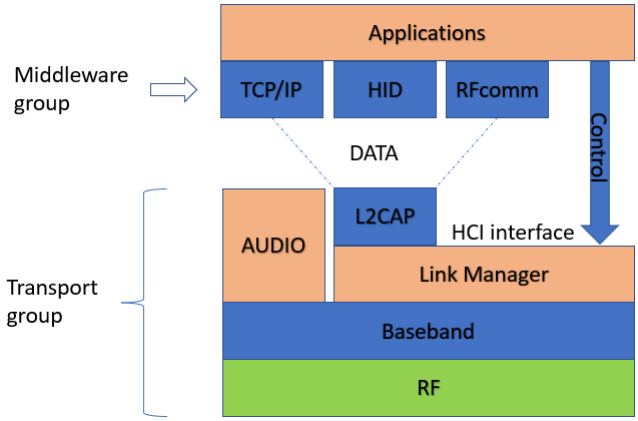

Figure 2: Bluetooth core Protocols groups.

Each device has a unique media access control address (MAC address) that identifies it on the network. Peripherals advertise this MAC address along with other information about the Peripheral's settings.

A Master may send and request data to a slave through something named a Characteristic. Each one belongs to a Service, which is like a container for the Characteristics. This paradigm is called the Bluetooth Generic Attribute Profile (GATT).

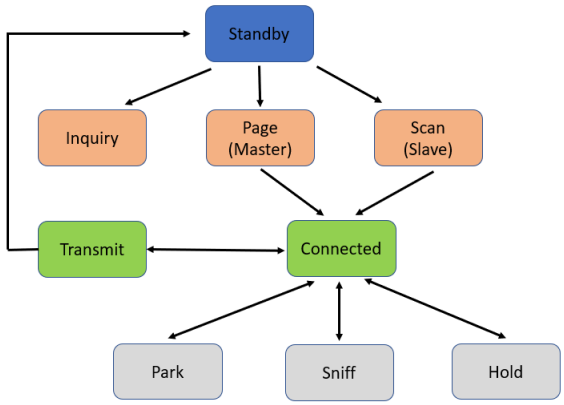

Figure 3: Bluetooth Device connection states.

The evolution of this protocol is shown in Table 1. This table enligths the main feature of each version and the year that was released.

Table 1: Bluetooth protocol evolution.

Year introduced	Bluetooth version	Feature
2004	2.0	Enhaced data rate
2007	2.1	Secure simple pairing
2009	3.0	High speed with 802.11 Wi-fi radio
2010	4.0	Low energy protocol
2013	4.1	Indirect IoT device connection
2014	4.2	IPv6 protocol for direct internet connection
2016	5.0	4x range, 2x speed, 8x message capacity + IoT

2.2 Bluetooth module description.

HC-05 is a class 2 (10m) Bluetooth Serial Port Protocol (SPP) module, designed for transparent wireless serial connection setup. The serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data

Rate) 3 Mbps modulation with a complete 2.4 GHz radio transceiver and baseband. See Figure 4.

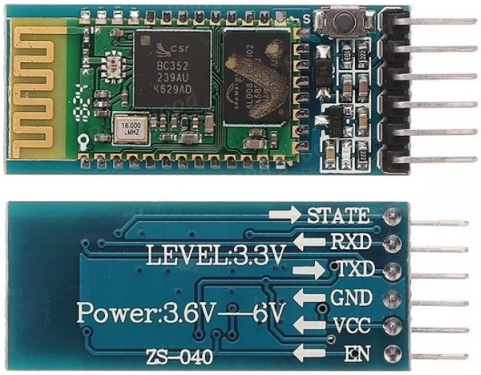

Figure 4: HC-05 Bluetooth module.

Some characteristics are:

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- Low Power Operation, 3.3 to 5 V.
- I/O Port input and output control.
- UART interface with programmable baud rate.
- Default Baud rate: 38400, Data bits:8, Stop bit:1, Parity: No parity.
- Supported baud rate: 9600,19200,38400, 57600,115200,230400,460800.

2.3 Microcontroller - Bluetooth connection.

The Bluetooth module is interconnected to a microcontroller unit (MCU) board for sending and receiving data from sensors or mobile applications respectively. For this, the following connections need to be done. VCC and GND pins are connected to 5 VCC and GND ports of the MCU board, respectively. The transmission(TX) and reception(RX) pins are connected to 10 and 11 pins configured as serial RX and TX in the Arduino platform. See Figure 5.

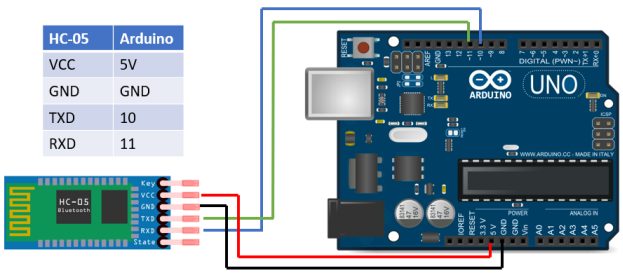

Figure 5: Wiring diagrams of HC-05 and Arduino Board.

Before the program is uploaded to the MCU, HC-05 parameters using AT commands are configured. For this, a straightforward program to connect the serial monitor to the Bluetooth module was written as shown in Figure 6.

```
#include <SoftwareSerial.h>

SoftwareSerial miBT(10,11);

void setup() {
  Serial.begin(9600);
  Serial.println("Listo");
  miBT.begin(38400);
}

void loop() {
  if (miBT.available())
    Serial.write(miBT.read());
if (Serial.available())
    miBT.write(Serial.read());
}
```

Figure 6: SW Code for AT configuration.

The Software Serial library configures pins 10 and 11 as RX and TX for sending data and commands to the HC-05 module.

In the setup part, the Serial Monitor is configured to 9600 bps and Universal asynchronous receiver-transmitter(UART) for Bluetooth communications to 38400 bps.

The program sends whatever is typed in the serial monitor to the Bluetooth transceiver, and the answer of this module is written to the serial monitor. Once the serial monitor is opened, the first step is to set the module in AT mode to configure its parameters. This mode is achieved with the following procedure:

Connect the MCU circuit to the PC USB port. In the beginning, the LED on the HC-05 is blinking fast, which means the VCC pin should be disconnected and press the button configuration. Reconnect the VCC while holding the button for five seconds, and then release it. If the LED on HC-05 starts to blink slowly, it means the module is now in configuration mode and can accept AT commands.

The serial monitor shows the LISTO message, then type AT, followed by enter. HC-05 must answer OK. This confirms that it is prepared to send AT commands. The most important commands are:
AT+NAME? gives module NAME: usually HC-05
AT+NAME=MIBT set MIBT name
AT + PSWD? Gives the 1234 pin code
AT+PSWD=5678 Set the pin code to 5678
AT+ROLE? Gives the Slave or Master role, zero or 1 , respectively.
AT+UART? Gives the speed, stop and parity parameters.
AT+ORGL restores the fabric parameters

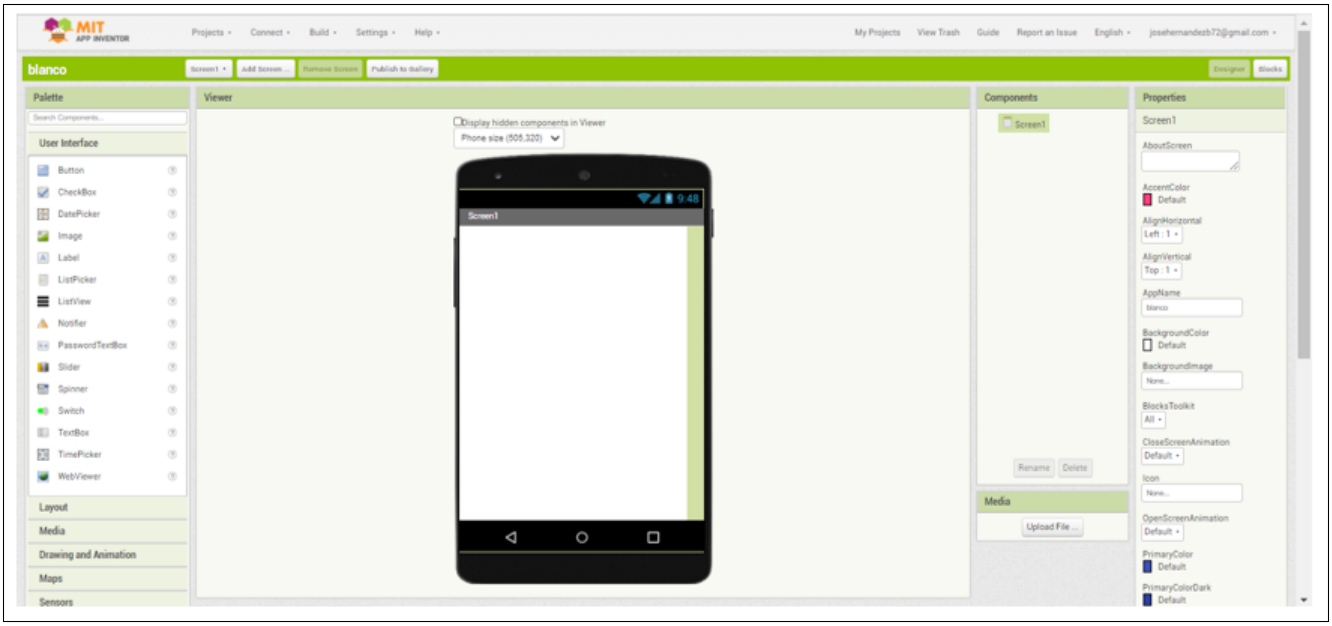

Figure 7: MIT App Inventor (Designer view).

AT+RESET get out of AT Mode

2.4 MIT Inventor application.

An application was developed for running on a mobile device with the Android or iOS operative system. MIT App Inventor is a development platform. This is composed of two parts: a web environment tool for programming in Java, Scheme, and Kawa, and an AppStore for downloading mobile equipment. The App Inventor is a block-oriented design tool, which allows a fast procedure for programming.

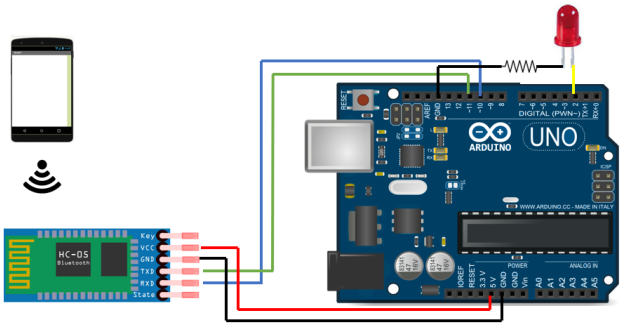

Figure 8: Led Control Application.

The program is first developed on a web page on the PC and then transferred to the mobile device through scan code authentication. The user interface has a block view for dragging and block connecting. Before transferring the application, the designer view shows the mobile user interface appearance for the application developed as illustrated in Figure 7.

3 Results

To illustrate the integration of the hardware and software part of the project. The developments of two applications are presented.

Test case 1: Led activation.

In this test case, a Led is connected to the Arduino's MCU digital port number 2 configured as an output, and the red Led is controlled by the application as illustrated in Figure 8. The software in the Arduino IDE is shown in Figure 9.

```
#include <SoftwareSerial.h>

SoftwareSerial BT(10,11);
char DATO=0;

void setup() {
  BT.begin(38400);
  pinMode(2,OUTPUT);
}

void loop() {
  if (BT.available())
  {
    DATO=BT.read();
    if (DATO=='r')
      digitalWrite(2,!digitalRead(2));
  }
 }
```

Figure 9: Arduino Code for application LED-control

The block coding built in MIT app Inventor scans the nearby devices for showing to the user. The second block reads the selection, connects, and sets the label to "connected". The third block reads the picking of the Toggle button to send the letter command, "r" in this case.

Test case 2: Potentiometer - Led application

In this test case, the program adds an analog voltage data acquisition for remote sensing on a mobile device. The voltage comes from a potentiometer, simulating the output of an analog sensor. The midpoint of the potentiometer is connected to the analog input A0 of the MCU. The software(SW) code is shown in Figure 10.

```
#include <SoftwareSerial.h>

SoftwareSerial BT(10,11);
char DATO=0;

void setup() {
  BT.begin(38400);
  pinMode(2,OUTPUT);
}

void loop() {
  if (BT.available())
  {
    DATO=BT.read();
    if (DATO=='r')
      digitalWrite(2,!digitalRead(2));
     }
  BT.print("#");
  BT.println(analogRead(A0));
   delay(20);
}
```

Figure 10: SW Code for Potentiometer-Led Application.

In the loop part, it has been noticed that if the Bluetooth received an 'r' character, the state of the red led changes. Meanwhile, the potentiometer lecture is taken in A0 port and sent to HC-05 preceded for a flag character % that the application detects as Boolean. The MIT App inventor code is shown in Figure 11.

4 Conclusions

A Bluetooth module was interconnected to an MCU board through an asynchronous serial port. Two applications were developed using the HC-05 and a mobile device. In the first test case scenario, the Led connected to one digital port was set on/off by the mobile application. In the second test, a Potentiometer was added, and the application in the mobile device monitored the received analog signal. The hardware and software employed in these monitoring applications showed the simplicity of building a powerful tool for rapid prototypes on IoT systems.

References

[Abdullah et al., 2022] Abdullah, M. I., Raya, L., Norazman, M. A. A., and Suprihadi, U. (2022). Covid-19 patient health monitoring system using iot. In *2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC)*, pages 155–158.

[Adiono et al., 2019] Adiono, T., Anindya, S. F., Fuada, S., Afifah, K., and Purwanda, I. G. (2019). Efficient android software development using mit app inventor 2 for bluetooth-based smart home. *Wireless Personal Communications*, 105:233–256.

[Bansal et al., 2021] Bansal, N., Jayant, K. P., Singh, P., and Pandey, S. (2021). Intelligent extension with smart connections using bluetooth with iot. In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 435–439.

[Collotta et al., 2018] Collotta, M., Pau, G., Talty, T., and Tonguz, O. K. (2018). Bluetooth 5: A concrete step forward toward the iot. *IEEE Communications Magazine*, 56(7):125–131.

[Kodali et al., 2020] Kodali, R. K., John, J., and Boppana, L. (2020). Iot monitoring system for grain storage. In *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6.

[Meruje, 2018] Meruje, M. (2018). *A Tutorial Introduction to IoT Design and Prototyping with Examples*, pages 153–190.

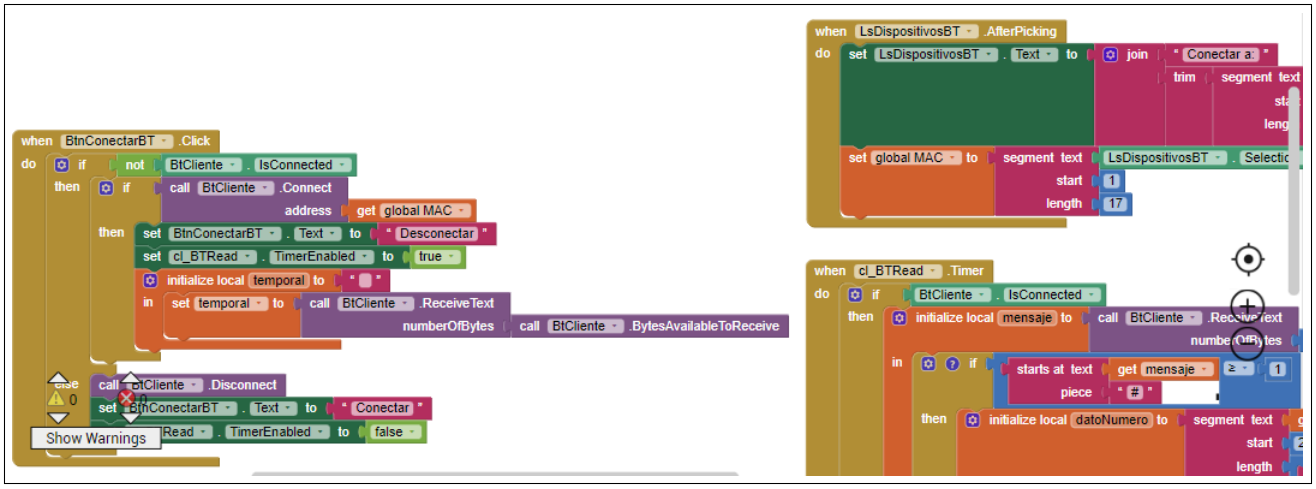

Figure 11: MIT App program.

[Mir and Llueca, 2020] Mir, S. B. and Llueca, G. F. (2020). Introduction to programming using mobile phones and mit app inventor. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 15(3):192–201.

[Mohammadi and Rashidzadeh, 2021] Mohammadi, F. and Rashidzadeh, R. (2021). An overview of iot-enabled monitoring and control systems for electric vehicles. *IEEE Instrumentation and Measurement Magazine*, 24(3):91–97.

[Munasinghe et al., 2019] Munasinghe, T., Patton, E. W., and Seneviratne, O. (2019). Iot application development using mit app inventor to collect and analyze sensor data. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6157–6159.

[St. Georgiev, 2019] St. Georgiev, T. (2019). Students' viewpoint about using mit app inventor in education. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 611–616.