UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE TECNOLOGÍA
CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

"CONTADOR DE MONEDAS BOLIVIANAS CONTROLADA POR UN APK MEDIANTE MODULO BLUETOOTH PARA EL SERVICIO DE TRANSPORTE PUBLICO DE LA CIUDAD DE LA PAZ"

Examen de Grado presentado para obtener el Grado de Licenciado en Electrónica y Telecomunicaciones

Presentado por: Dither Ariel Urquieta Guarachi

La Paz - Bolivia
Agosto, 2018

DEDICATORIA

Este trabajo de aplicación va dedicado a todos mis seres queridos y amados. A mis padres, tíos, abuelos, amigos de colegios y a mis estudiantes que siempre seré para ellos un gran ejemplo de persona que admiran, respetan y sobre todo quieren mucho.

AGRADECIMIENTO

Agradecido a mis padres que me apoyaron en el desarrollo de mis estudios desde que era un bebé, a mis profesores, docentes e estudiantes que hicieron que mi conocimiento crezca y se fortalezca hasta formar a la persona que soy hoy en día, ellos fueron la fuente de energía que hicieron que se cumplan mis logros y metas y estarán para seguir apoyándome para lo que me depara el futuro…

RESUMEN DEL TRABAJO

Hoy en día todos queremos que las maquinas hagan nuestro trabajo de forma más rápida y eficiente, es por eso que contar monedas nos puede llevar tiempo y para evitar eso, podemos hacer que un equipo electrónico se encargue de realizar esa operación, además de clasificar las monedas para así tenerlas por separadas en caso que se quiera utilizar nuevamente las mismas.

Uno de los empleos donde se genera una gran cantidad de monedas es en el servicio de transporte público donde los usuarios normalmente pagan con monedas fraccionarias.

Este proyecto consiste en identificar el valor correspondiente de cada moneda y la sumatoria total de totas las monedas que nuestro equipo vaya a clasificar.

Para eso se utilizara una aplicación android donde se registrara la cantidad de monedas que se contó mediante un sistema de conteo que lo realizara un PIC16F877A y la comunicación se hará a través del módulo bluetooth.

Con el microcontrolador PIC16F877A se realiza un detector de monedas, que nos indica el valor de la moneda que pasó por el sistema.

Gracias a un módulo bluetooth, lograremos la comunicación desde el PIC hacia un dispositivo Android mediante la comunicación serial Uart.

Se desarrolla un software en App Inventor en donde se realiza un diseño para que el sistema de conteo pueda ser monitoreado a través de un dispositivo Android.

INDICE

DEDICATORIA.....ii
AGRADECIMIENTO.....iii
RESUMEN DEL TRABAJO.....iv
CAPÍTULO I.....7
1. PLANTEAMIENTO DEL PROBLEMA.....7
1.1. IDENTIFICACIÓN DEL PROBLEMA.....7
1.2. JUSTIFICACIÓN DE TRABAJO.....8
1.3. OBJETIVOS.....8
2. FUNDAMENTO TEÓRICO.....10
2.1. Contador y separador de monedas.....10
2.2. Microcontrolador PIC16F877A.....10
2.3. Arquitectura del microcontrolador.....11
Figura Nº1. Arquitectura del microcontrolador.....11
Figura Nº 2: Encapsulado DIP del PIC 16F877 y la distribución de sus 40 pines.....12
Figura Nº 3. Los pines I/O (Input/Output).....13
2.3.1. Arquitectura interna.....15
Figura Nº 5: Arquitectura interna.....15
2.3.2. Estructura interna del PIC 16F877......16
2.3.3. Memoria de programa (FLASH).....16
Figura Nº 6: Arquitectura interna.....16
2.3.4. Mapa de memoria de programa (FLASH)......17
2.3.5. Memorias de datos.....18
2.3.5.1. Memoria SRAM (Static Random Access Memory):.....18
Figura Nº7: Tabla de registros del PIC 16F877 y sus direcciones.....18
2.3.5.2. Memoria EEPROM:.....19
2.3.5.3. Reloj u Oscilador.....19
FIGURA Nº 8: configuración en XT y RC.....20
2.4. Modulo bluetooth HC-05.....21
FIGURA Nº 9: Módulo Bluetooth.....21

2.5. App Inventor.....23
FIGURA Nº10: App Inventor diseño y bloques 23
Figura Nº11 Página de inicio de AppInventor. 24
Figura Nº12: Página para la creación de proyectos. 24
Figura Nº13: Creación del nuevo proyecto. 25
Figura Nº14: Estructura del entorno de edición. 26
CAPÍTULO III.....27
3. DESARROLLO DEL PROYECTO 27
3.1. Diagrama de flujo.....27
Figura Nº1 Diagrama de flujo del sistema de conteo 27
Figura Nº2 Diagrama de bloques del sistema de conteo.....28
3.3. Diagrama del circuito.....29
Figura Nº3 Diagrama del circuito del sistema de conteo.....29
3.4. Diagrama de la aplicación.....30
Figura Nº4 Diagrama de la aplicacion del sistema de conteo 30
3.5. Programación del contador en MikroBasic.....31
3.6. Programación del contador en AppInventor2.....32
3.6.1. Conexión y desconexión con el modulo bluetooth HC-05.....33
3.6.2. Contador de monedas.....33
3.6.3. Nuevo Conteo.....34
CAPÍTULO IV.....36
4. CONCLUSIONES.....36
CAPÍTULO V 37
5. BIBLIOGRAFIA.....37
ANEXOS.....38
PIC16F877A Pin Configuration.....38
PIC16F877A Features.....39

CAPÍTULO I

1. PLANTEAMIENTO DEL PROBLEMA

1.1. IDENTIFICACIÓN DEL PROBLEMA

El tiempo uno de los recursos más valiosos en nuestras vidas que no se puede recuperar, hace que las necesidades de tener tecnologías avanzadas en nuestra sociedad.

En todo negocio surge la necesidad de hacer cambio de productos a cambio de dinero, para esto nuestra medida de cambio son los billetes y monedas de estado plurinacional de Bolivia.

En muchos de estos negocios existe una inmensa cantidad de monedas en donde la suma de todas esas nos da la ganancia que existe en el negocio.

En Bolivia existe un montón de empleos en donde te pagan normalmente con monedas, esto hace que una persona se encargue de contar estas monedas para ver el monto total generado, pero esto se vuelve muy relevante cuanto las monedas excesivamente altas, haciendo que el contador demore en hacer esta suma monetaria.

Debido al agotamiento físico y mental por la cantidad excesiva de monedas hace que uno pueda fallar en el conteo monetario, que viendo desde otro punto de vista hace que puede existir error de suma de dinero, que incluso puede llegar a perjudicar a muchos empleados que realicen a estos tipos de trabajo.

1.2. JUSTIFICACIÓN DE TRABAJO

En el servicio de transporte Público de la ciudad de La Paz hay muchas personas que hoy en día siguen pagando su pasaje en efectivo.

En estos casos en la tarifa del pasaje es de 2.00 Bs y en horario nocturno 2.20 Bs, esto hace que muchos usuarios paguen con monedas fraccionarias.

La acumulación de todo un día de trabajo hace que se genere una enorme cantidad de monedas y esta aumenta a medida que pasa los días.

La necesidad de contar las ganancias realizadas en un día, una semana o de un mes en una empresa de transporte, nos da un informe de datos estadísticos de cómo son las ganancias diarias del servicio de transporte además de un control de manejo de cambio evitara que no exista perdidas monetarias.

Es por eso que empleamos con una nueva tecnología que hace que este trabajo sea preciso y eficiente, garantizando el control de las ganancias generadas en un día de trabajo.

1.3. OBJETIVOS

1.3.1. General

Diseñar un sistema de conteo de monedas clasificadas de acuerdo a su valor monetario para realizar el control de ganancias generadas en el servicio de transporte público de la ciudad de La Paz.

1.3.2. Específicos

- Desarrollar el Hardware del sistema utilizando el microcontrolador PIC16F887A.
- Desarrollar el Software del sistema de conteo para poder detectar las monedas a contar.
- Configurar el puerto serial para poder realizar la comunicación serial entre el PIC16F877A y un dispositivo Android.

- Utilizar el AppInvertor2 para poder realizar el monitoreo de sistema de conteo.
- Desarrollar la programación en App Inventor para poder comunicarnos con el modulo bluetooth.
- Desarrollar un software en App Inventor para obtener los resultados del sistema de conteo.

CAPÍTULO II

2. FUNDAMENTO TEÓRICO

2.1. Contador y separador de monedas

Las contadoras de monedas son separadoras de monedas en tiempo real y funcionan contabilizando y clasificando el dinero. Cada tipo de moneda, es separada y clasificada. Estos dispositivos funcionan con una bandeja en la parte superior y otra la cual tiene orificios que clasifica las monedas en función de su tamaño y otras características para contadores un poco más tecnológicos.

Las máquinas contadoras de monedas comenzaron siendo utilizadas inicialmente por entidades bancarias para contar y clasificar monedas de forma eficiente. Poco a poco los contadores de monedas empezaron a popularizarse y acabaron por llegar a todas las entidades que manejan dinero como supermercados y otros negocios. Un contador y clasificador de monedas ahorra mucho tiempo ya que el trabajo de separar y contar las monedas puede ser tedioso y demorado.

2.2. Microcontrolador PIC16F877A

Se denomina microcontrolador a un dispositivo programable capaz de realizar diferentes actividades que requieran del procesamiento de datos digitales y del control y comunicación digital de diferentes dispositivos. Los microcontroladores poseen una memoria interna que almacena dos tipos de datos; las instrucciones, que corresponden al programa que se ejecuta, y los registros, es decir, los datos que el usuario maneja, así como registros especiales para el control de las diferentes funciones del microcontrolador. Los micro controladores se programan en Assembler y cada microcontrolador varía su conjunto de instrucciones de acuerdo a su fabricante y modelo, de acuerdo al número de instrucciones que el microcontrolador maneja se le denomina de arquitectura RISC (reducido) o CICS (complejo). Los microcontroladores poseen principalmente una

ALU (Unidad Lógico Aritmética), memoria del programa, memoria de registros, y pines I/O (entrada y/o salida). La ALU es la encargada de procesar los datos dependiendo de las instrucciones que se ejecuten (ADD, OR y AND), mientras que los pines son los que se encargan de comunicar al microcontrolador con el medio eterno; la función de los pines puede ser de transmisión de datos, alimentación de corriente para el funcionamiento de este o pines de control especifico. En este proyecto se utilizó el PIC16F877A. Este microcontrolador es fabricado por MicroChip familia a la cual se le denomina PIC. El modelo 16F877 posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y práctico para ser empleado en la aplicación que posteriormente será detallada. (Hector Morlote, 2012-2013)

2.3. Arquitectura del microcontrolador

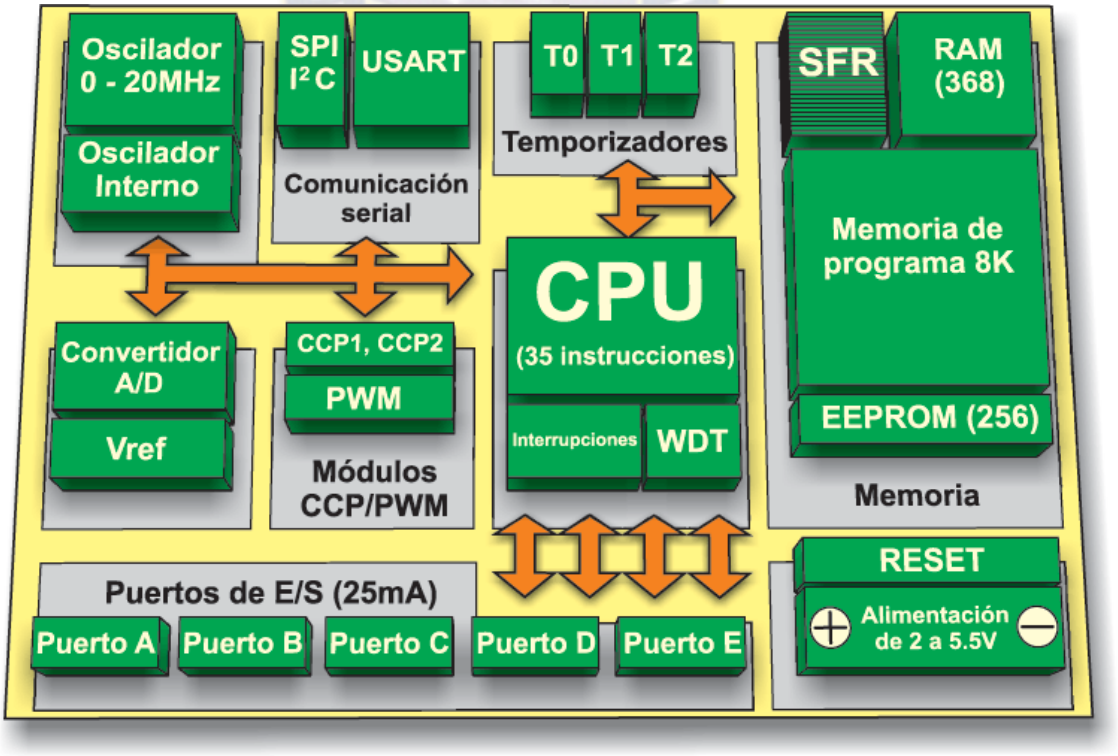


Figura Nº1. Arquitectura del microcontrolador
Fuente: https://es.slideshare.net/alandbravo/2014-ii-c08tsbc-pic-para-ecg

- Memoria de programa: FLASH de 8K de instrucciones de 14 bits
- Memorias de datos: SRAM de 512 bytes, EEPROM de 256 bytes
- Pines I/O (Input/Output) : 6 del puerto A, 8 del puerto B, 8 del puerto C, 8 del puerto D y 3 del puerto E, además de 8 entradas analógicas.
- Pila (Stack): 8 niveles (14 bits)
- Fuentes de interrupción: 14
- Instrucciones: 35
- Compatible modo SLEEP
- Frecuencia máxima del oscilador de 20MHz
- Conversor Analógico/Digital de 10 bits multicanal (8 canales de entrada)
- Corriente máxima absorbida/suministrada (sink/source) por pin: 25 mA
- Voltaje nominal: 3 a 5.5V DC (CMOS)
- Power On Reset
- Power Up Timer (PWRT)
- Oscilador Start Up Timer (OST)

El encapsulado que he utilizado es de tipo DIP (*Dual In-Line Pin)* de 40 pines, aunque posee otros encapsulados (SOIC, PLCC y QFP):

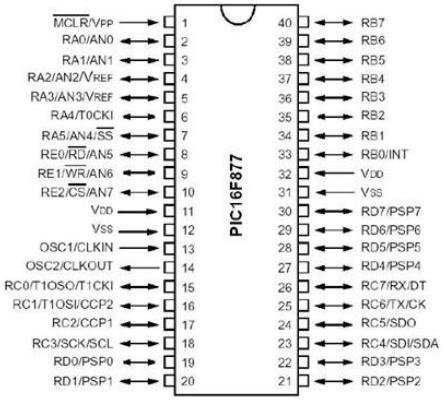


Figura Nº 2: Encapsulado DIP del PIC 16F877 y la distribución de sus 40 pines.
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

Los pines I/O (Input/Output) están organizados en 5 puertos:

- Puerto A: 6 pines
- Puerto B: 8 pines
- Puerto C: 8 pines
- Puerto D: 8 pines
- Puerto E: 3 pines

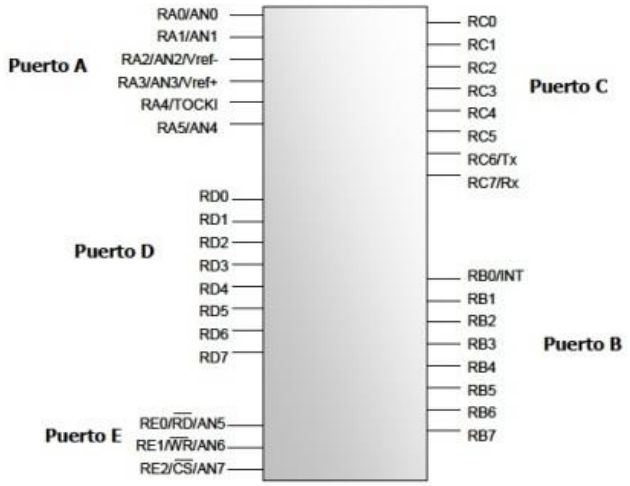


Figura Nº 3. Los pines I/O (Input/Output)
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

Cada pin de esos puertos se puede configurar como entrada o como salida independiente programando un par de registros diseñados para tal fin. En ese registro un bit en "0" configura el pin del puerto correspondiente como salida y un bit en "1" lo configura como entrada. Dichos pines del microcontrolador también pueden cumplir otras funciones especiales, siempre y cuando se configuren para ello. En la siguiente tabla se indican las funciones de todos los pines del PIC:

Nombre pin	Pin	Descripción
RA0/AN0	2	E/S Digital o Entrada análoga 0.
RA1/AN1	3	E/S Digital o Entrada análoga 1.
RA2/AN2 V_{ref} -	4	E/S Digital o Entrada análoga 2.
RA3/AN3/V_{ref} +	5	E/S Digital o Entrada análoga 3.
RA4/T0CKI	6	Bit 4 del puerto A (E/S bidireccional). También se usa como entrada de reloj al temporizador/contador TMR0. Salida de colector abierto.
RA5/SS/AN4	7	E/S Digital o Entrada análoga 4. También lo usa el puerto serial síncrono.
RB0/INT	33	Bit 0 del puerto B (E/S bidireccional). Buffer E/S: TTL/ST. También se usa como entrada de interrupción externa (INT).
RB1	34	Bit 1 del puerto B (E/S bidireccional). Buffer E/S: TTL
RB2	35	Bit 2 del puerto B (E/S bidireccional). Buffer E/S: TTL
RB3/PGM	36	Bit 3 del puerto B (E/S bidireccional). Buffer E/S: TTL (Programación en bajo voltaje)
RB4	37	Bit 4 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio del pin.
RB5	38	Bit 5 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio del pin.
RB6/PGC	39	Bit 6 del puerto B (E/S bidireccional). Buffer E/S: TTL/ST. Interrupción por cambio del pin. Entrada de reloj para programación serial.
RB7/PGD	40	Bit 7 del puerto B (E/S bidireccional). Buffer E/S: TTL/ST. Interrupción por cambio del pin. Entrada de datos para programación serial.
RC0/T1OSO/T1CKI	15	E/S Digital. Salida del oscilador Timer 1 o entrada de reloj Timer 1.
RC1/T1OSI/CCP2	16	E/S Digital. Entrada del oscilador Timer 1. Entrada Captura 2; Salida Compara 2; Salida PWM 2
RC2/CCP1	17	E/S Digital. Entrada Captura 1; Salida Compara 1; Salida PWM 1
RC3/SCK/SCL	18	E/S Digital. Línea de reloj serial asíncrono en el modo SPI y el modo I^2C
RC4/SDI/SDA	23	E/S Digital. Línea de datos en el modo SPI o en el modo I^2C
RC5/SDO	24	E/S Digital.
RC6/TX/CK	25	E/S Digital. Transmisión asíncrona (USART) o reloj síncrono (SSP).
RC7/RX/DT	26	E/S Digital. Recepción asíncrona (USART) o línea de datos (SSP).
V_{DD}	11,32	Voltaje de alimentación DC (+)
V_{SS}	12,31	Referencia de voltaje (GND).
MCLR	1	Entrada de RESET al microcontrolador. Voltaje de entrada durante la programación. En nivel bajo resetea el microcontrolador.
OSC1/CLKIN	13	Entrada oscilador cristal oscilador / Entrada fuente de reloj externa.
OSC2/CLKOUT	14	Salida oscilador cristal. Oscilador RC: Salida con un ¼ frecuencia OSC1
RD0/PSP0	19	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD1/PSP1	20	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD2/PSP2	21	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD3/PSP3	22	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD4/PSP4	27	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD5/PSP5	28	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD6/PSP6	29	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD7/PSP7	30	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RE0/RD/AN5	8	E/S Digital. Puede se pin de lectura (*read*) en modo microprocesador.
RE1/WR/AN6	9	E/S Digital. Puede ser pin de escritura (*write*) en modo microprocesador.
RE2/CS/AN7	10	E/S Digital. Puede ser pin de selección de chip (*chip select*) en modo microprocesador.

Figura Nº 4: En la tabla superior, donde aparecen las siglas "E/S" hacen referencia a "Entrada/Salida", lo mismo que "I/O" (Input/Output).

Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2.3.1. Arquitectura interna

Este término se refiere a los bloques funcionales que componen en PIC internamente, como la memoria RAM, la memoria FLASH, la lógica de control, etc. (Hector Morlote, 2012-2013)

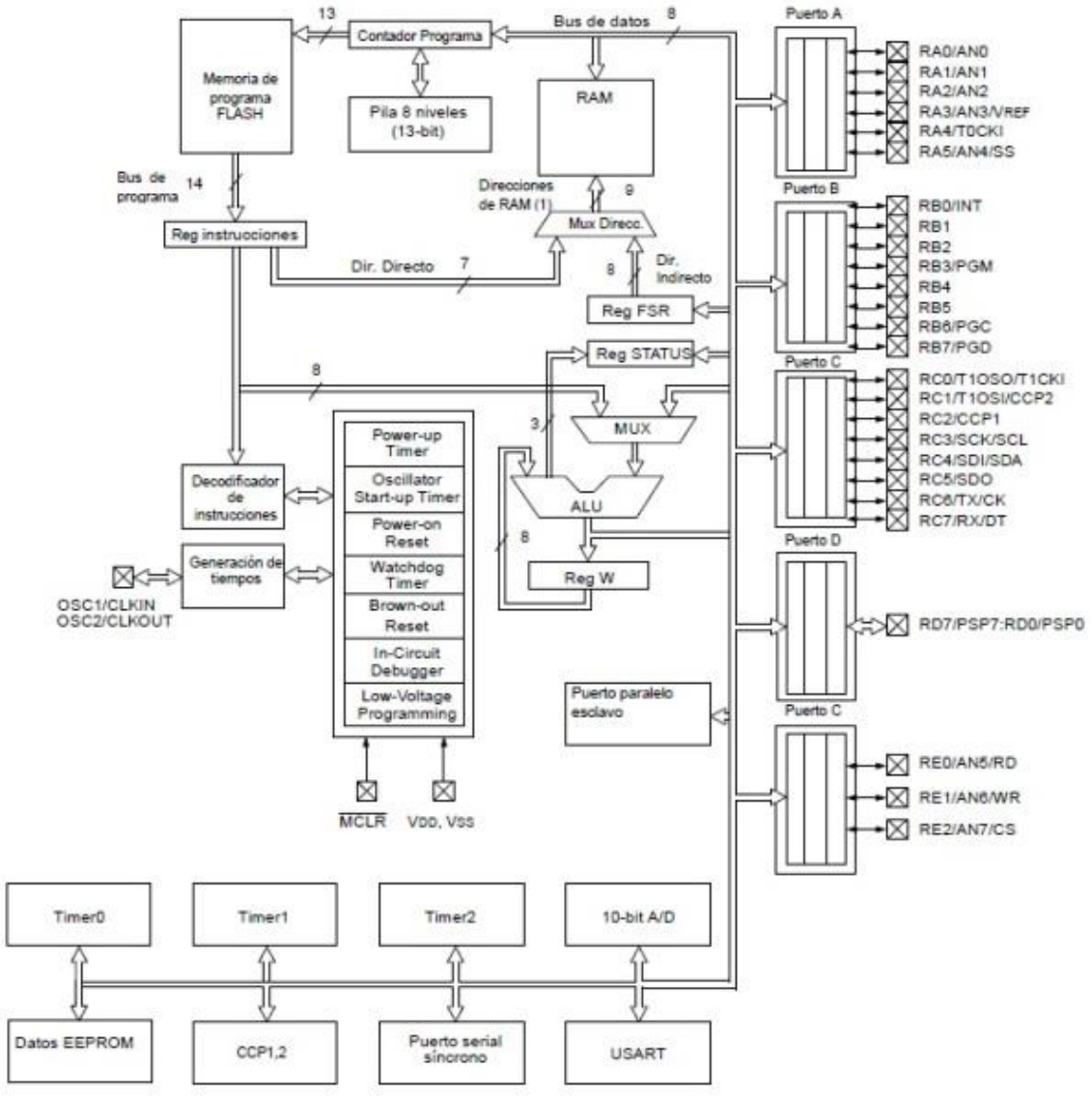


Figura Nº 5: Arquitectura interna
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2.3.2. Estructura interna del PIC 16F877.

El PIC 16F877 se basa en la arquitectura Harvard, en la cual el programa y los datos se pueden trabajar con buses (un bus es un conjunto de líneas que transportan información entre 2 o más módulos) y memorias separadas, lo cual permite que las instrucciones y los datos tengan longitudes diferentes. Hector Morlote (2012-2013)

2.3.3. Memoria de programa (FLASH)

Es una memoria de 8K de capacidad con posiciones de 14 bits. En ella se graba o almacena el programa o códigos que el microcontrolador debe ejecutar.(Hector Morlote, 2012-2013)

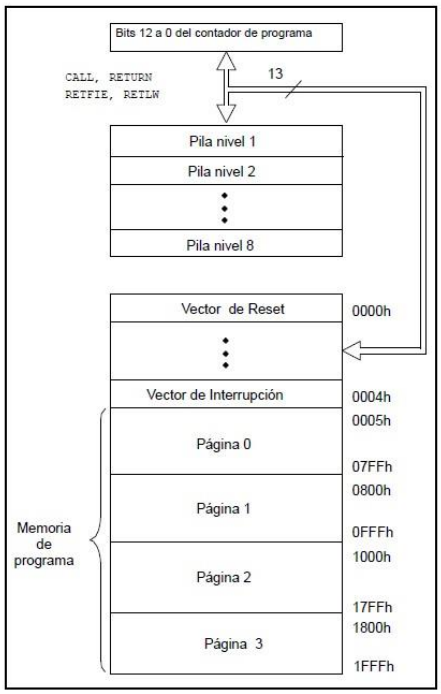


Figura Nº 6: Arquitectura interna
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2.3.4. Mapa de memoria de programa (FLASH).

- La memoria está dividida en cuatro páginas de 2K cada una. La Página 0 va de la posición de memoria 0005h a la 07FFh, la Página 1 de 0800h a 0FFFh, la Página 2 de 1000h a 17FFh y la Página 3 de 1800h a 1FFFh.

- El contador de programa (en este caso es de 13 bits) nos indica la dirección de la instrucción a ejecutar.

- *Pila (Stack)*: son registros que no forman parte de ningún banco de memoria (los bancos de memoria los explico más abajo) y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuandose hace un llamado a una subrutina o a una interrupción. Cuando el micro vuelva a ejecutar su tarea normalmente, el contador de programa recupera su valor leyéndolo en la pila. Al tener una pila de 8 niveles, se pueden acumular 8 llamadas a subrutinas sin tener problemas.

- *Vector de RESET:* cuando se resetea el microcontrolador el contador de programa se pone a cero (0000h). Por esto, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

- *Vector de Interrupción:* cuando el microcontrolador recibe una llamada a una interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso allí se debe escribir toda la información necesaria para atender dicha interrupción. Hector Morlote (2012-2013)

2.3.5. Memorias de datos

Como he mencionado anteriormente, el PIC 16F877 tiene 2 memorias de datos:

2.3.5.1. Memoria SRAM (Static Random Access Memory):

Es una memoria de tipo volátil (cuando deja de recibir alimentación se borran los datos que tenga almacenados) que está dividida en 4 bancos de 128 bytes cada uno. De esos 128 bytes los 32 primeros están dedicados a los SFR´s (*Registros de Funciones Especiales,* cumplen un propósito general en el control y configuración del microcontrolador) y los 96 siguientes a los GPR´s (Registros *de Propósitos* Generales, se pueden usar para guardar los datos temporales de la tarea que se está ejecutando). *(Hector Morlote, 2012-2013)*

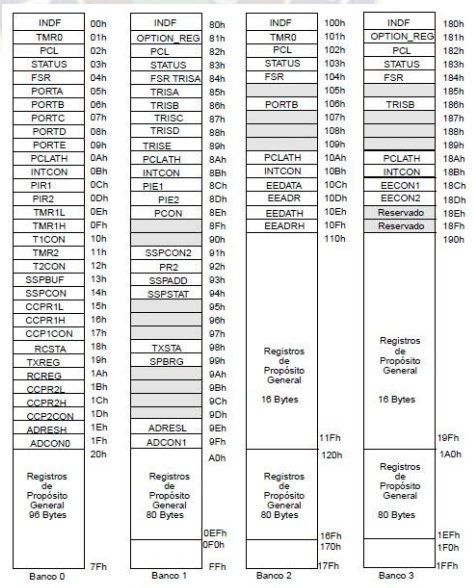


Figura Nº7: Tabla de registros del PIC 16F877 y sus direcciones.
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2.3.5.2. Memoria EEPROM:

Es una memoria no volátil (guarda los datos aunque le falte alimentación) con una capacidad de 256 bytes, que permite realizar operaciones de lectura y escritura sin interferir con el funcionamiento normal del microcontrolador. *(Hector Morlote, 2012-2013)*

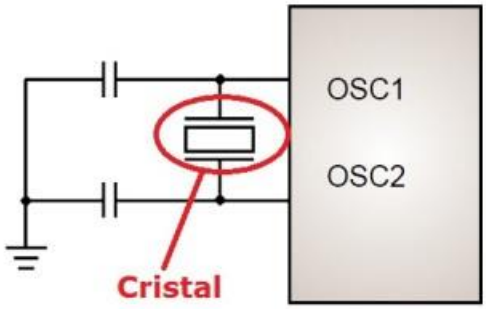
2.3.5.3. Reloj u Oscilador

El pequeño ciruito externo que los microcontroladores necesitan para que se les indique la velocidad de trabajo es conocido como reloj u oscilador. En función del montaje que se realice se puede conseguir más o menos precisión. En el momento de programar (o quemar los fusibles) el PIC se debe especificar el tipo de oscilador externo que se va a utilizar. El PIC 16F877 puede utilizar 4 tipos de oscilador diferentes:

- XT: Cristal genérico (de 1 a 4 MHz).
- RC: Oscilador con resistencia y condensador.
- HS: Cristal de alta frecuencia (de 10 a 20 MHz).
- LP: Cristal para baja frecuencia y bajo consumo.

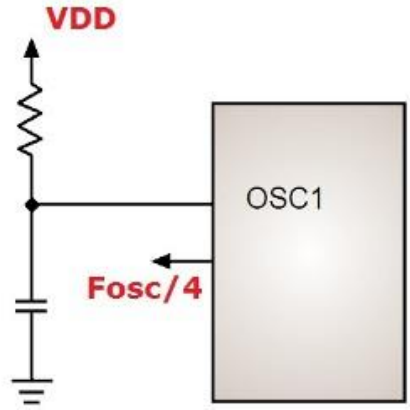
Las configuraciones más utilizadas son la XT y RC:

-XT: se suele utlizar con un cristal de 4 MHz, pues garantiza precisión y es bastante comercial. Internamente esta frecuencia es dividida entre 4, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz en este caso, por lo que cada instrucción se ejecuta en 1 μs (1 microsegundo):



Oscilador XT: el cristal debe ir acompañado de 2 condensadores.

-RC: se utiliza si no se precisa una gran precisión y se quiere economizar dinero:



Oscilador RC: sólo se necesita una resistencia y un condensador.

FIGURA Nº 8: configuración en XT y RC
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2.4. Modulo bluetooth HC-05

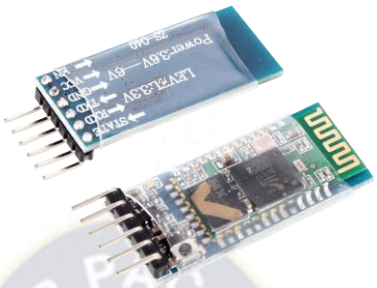


FIGURA Nº 9: Módulo Bluetooth

Fuente: http://itronsrobotics.com/producto/modulo-bluetooth-hc-05/

El módulo bluetooth HC-05 viene configurado de fábrica para trabajar como maestro o esclavo. En el modo maestro puede conectarse con otros módulos bluetooth, mientras que en el modo esclavo queda a la escucha peticiones de conexión. Agregando este módulo a tu proyecto podrás controlar a distancia desde un celular o una laptop todas las funcionalidades que desees.

El modulo BlueTooth HC-05 utiliza el protocolo UART RS 232 serial. Es ideal para aplicaciones inalámbricas, fácil de implementar con PC, microcontrolador o módulos Arduinos. *(Geek Factory, 2014)*

La tarjeta incluye un adaptador con 6 pines de fácil acceso para uso en protoboard.

Los pines de la board correspondientes son:

- EN
- VCC
- GND
- TXw
- RX
- STATE

Además posee un regulador interno que permite su alimentación de 3.6 a 6V.

Características:

- Especificación bluetooth v2.0 + EDR (Enhanced Data Rate)
- Puede configurarse como maestro, esclavo, y esclavo con autoconexión (Loopback) mediante comandos AT
- Chip de radio: CSR BC417143
- Frecuencia: 2.4 GHz, banda ISM
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Antena de PCB incorporada
- Potencia de emisión: ≤ 4 dBm, Clase 2
- Alcance 5 m a 10 m
- Sensibilidad: ≤ -84 dBm a 0.1% BER
- Velocidad: Asincrónica: 2.1 Mbps (max.)/160 kbps, sincrónica: 1 Mbps/1 Mbps
- Seguridad: Autenticación y encriptación (Password por defecto: 1234)
- Perfiles: Puerto serial Bluetooth
- Módulo montado en tarjeta con regulador de voltaje y 6 pines suministrando acceso a VCC, GND, TXD, RXD, KEY y status LED (STATE)
- Consumo de corriente: 50 mA
- El pin RX del módulo requiere resistencia de pull-up a 3.3 V (4.7 k a 10 k). Si el microcontrolador no tiene resistencia de pull-up interna en el pin Tx se debe poner externamente.
- Niveles lógicos: 3.3 V. Conectarlos a señales con voltajes mayores, como por ej. 5 V, puede dañar el módulo
- Voltaje de alimentación: 3.6 V a 6 V
- Dimensiones totales: 1.7 cm x 4 cm aprox.
- Temperatura de operación: -20 ºC a +75 ºC

2.5. App Inventor

App Inventor es un entorno de desarrollo de aplicaciones para dispositivos Android. Para desarrollar aplicaciones con App Inventor sólo necesitas un navegador web y un télefono o tablet Android (si no lo tienes podrás probar tus aplicaciones en un emulador). App Inventor se basa en un servicio web que te permitirá almacenar tu trabajo y te ayudará a realizar un seguimiento de sus proyectos.

Se trata de una herramienta de desarrollo visual muy fácil de usar, con la que incluso los no programadores podrán desarrollar sus aplicaciones. Al construir las aplicaciones para Android trabajarás con dos herramientas: App Inventor Designer y App Inventor Blocks Editor. En Designer construirás el Interfaz de Usuario, eligiendo y situando los elementos con los que interactuará el usuario y los componentes que utilizará la aplicación. En el Blocks Editor definirás el comportamiento de los componentes de tu aplicación. *(Carlos González, 2011)*

FIGURA Nº10: App Inventor diseño y bloques
Fuente: https://sites.google.com/site/appinventormegusta/primeros-pasos

El primer paso para la creación de la App será iniciar el entorno de trabajo. Para ello abre tu navegador de internet habitual y dirígete a la siguiente URL: http://appinventor.mit.edu

Verás cómo se carga la página de inicio de la aplicación, que en la parte inferior incluye tres botones: Teach, Explore e Invent, que dan acceso a recursos para los educadores, información y tutoriales, y a la creación de aplicaciones móviles, respectivamente. Puesto que nuestra intención es crear una aplicación, pulsaremos sobre el botón Invent. *(Carlos González, 2011)*

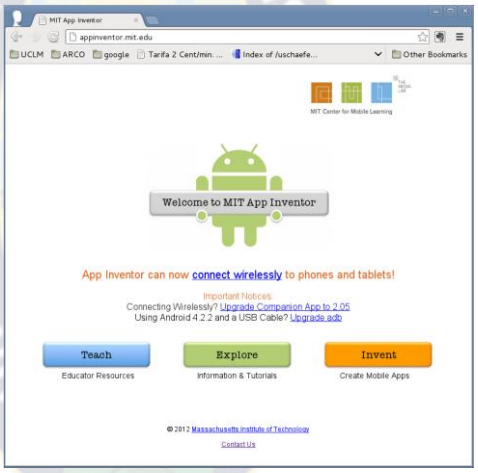


Figura Nº11 Página de inicio de AppInventor.
Fuente: http://webpub.esi.uclm.es/img/upload/plugin/ESI-TechLab-AppInventor2-2015beta.pdf

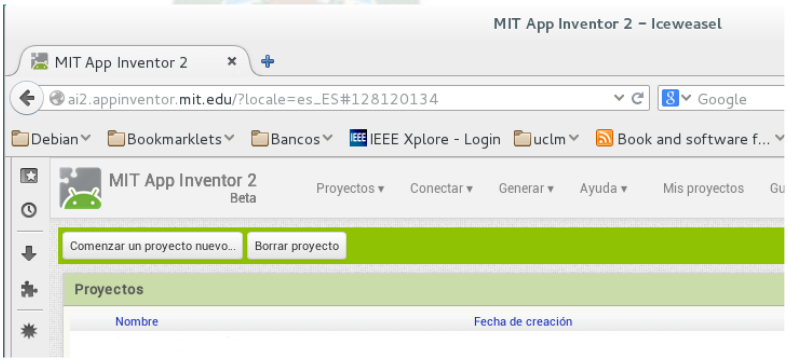


Figura Nº12: Página para la creación de proyectos.
Fuente: http://webpub.esi.uclm.es/img/upload/plugin/ESI-TechLab-AppInventor2-2015beta.pdf

AppInventor utiliza las cuentas de Google para el inicio de sesión, por lo que si llegados a este punto aun no la has iniciado, este será el momento de hacerlo. Si no dispones de una cuenta de Google puedes crearla también. A partir del momento en el que estés registrado, la web de AppInventor recordará todos tus proyectos cada vez que retornes a ella. Puesto que aun no hemos comenzado el desarrollo, la lista de proyectos estará vacía (figura 12).

Para comenzar a desarrollar la aplicación el primer paso consistirá en crear un nuevo proyecto. Para ello pulsa el botón Comenzar un proyecto nuevo... (figura 13), e introduce en el cuadro de diálogo que aparecerá a continuación el nombre del proyecto, que en este caso podría ser el siguiente: hola_programador. . *(Carlos González, 2011)*

Figura Nº13: Creación del nuevo proyecto.
Fuente: http://webpub.esi.uclm.es/img/upload/plugin/ESI-TechLab-AppInventor2-2015beta.pdf

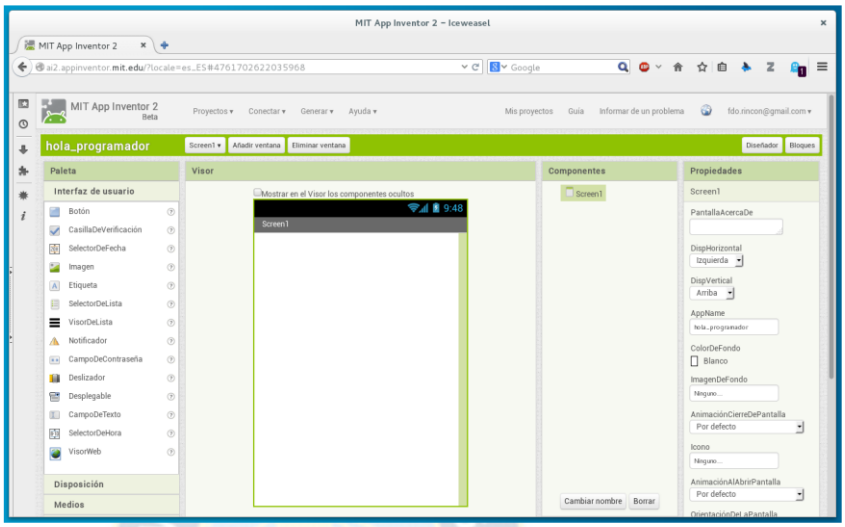


Figura Nº14: Estructura del entorno de edición.
Fuente: http://webpub.esi.uclm.es/img/upload/plugin/ESI-TechLab-AppInventor2-2015beta.pdf

Una vez creado el proyecto, aparecerá una nueva página, con un aspecto como el de la figura 14. Como puedes apreciar, en el centro de la página se dibuja la pantalla de un teléfono móvil. Sobre esta zona se colocarán los diferentes componentes gráficos de la aplicación, que se mostrarán con el aspecto que realmente tendrán en el dispositivo real.

En la parte izquierda de la pantalla se encuentra la paleta de componentes que pueden ser utilizados en la aplicación, organizados por categorías. Para este primer proyecto nos bastará con las que aparecen dentro de la categoría Interfaz de usuario.

En la parte derecha de la ventana hay 2 columnas más. La más próxima a la pantalla del móvil contiene la lista de los componentes que se han añadido a la aplicación. La otra muestra las propiedades (características) de aquel componente que esté seleccionado. Estas características sirven, por ejemplo, para cambiar el texto que aparece en el botón, cargar el fichero mp3 asociado a un sonido, etc. *(Carlos González, 2011)*

CAPÍTULO III

3. DESARROLLO DEL PROYECTO

En este proyecto lo que se quiere mostrar es como se realiza el conteo de monedas de diferentes valores

3.1. Diagrama de flujo

En el siguiente diagrama se ve paso a paso que es lo que realiza el contador de monedas

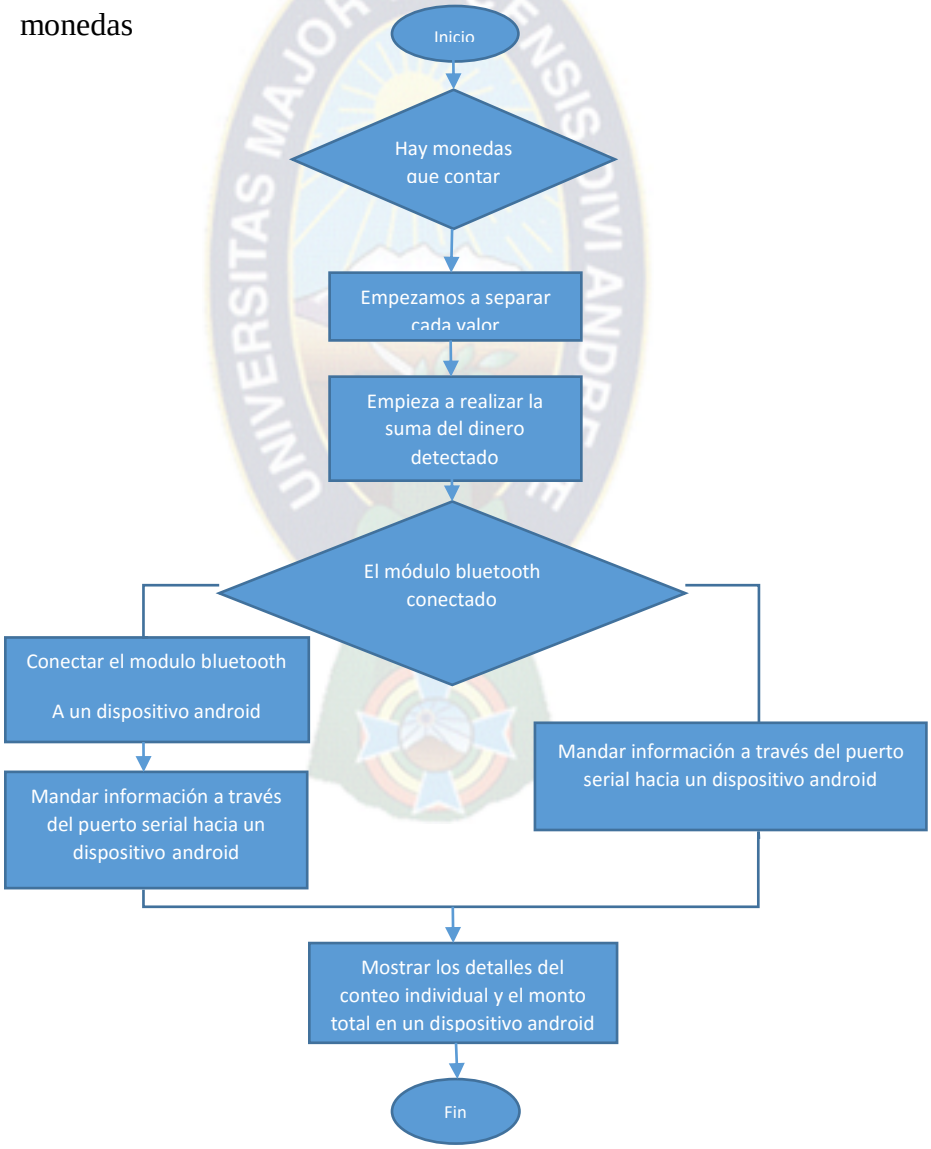


Figura Nº1 Diagrama de flujo del sistema de conteo
Fuente: Propia

3.2. Diagrama en bloques:

En el diagrama se ve la arquitectura de cómo está armado el sistema, donde se ve a un microcontrolador PIC comunicarse con todos sus periféricos.

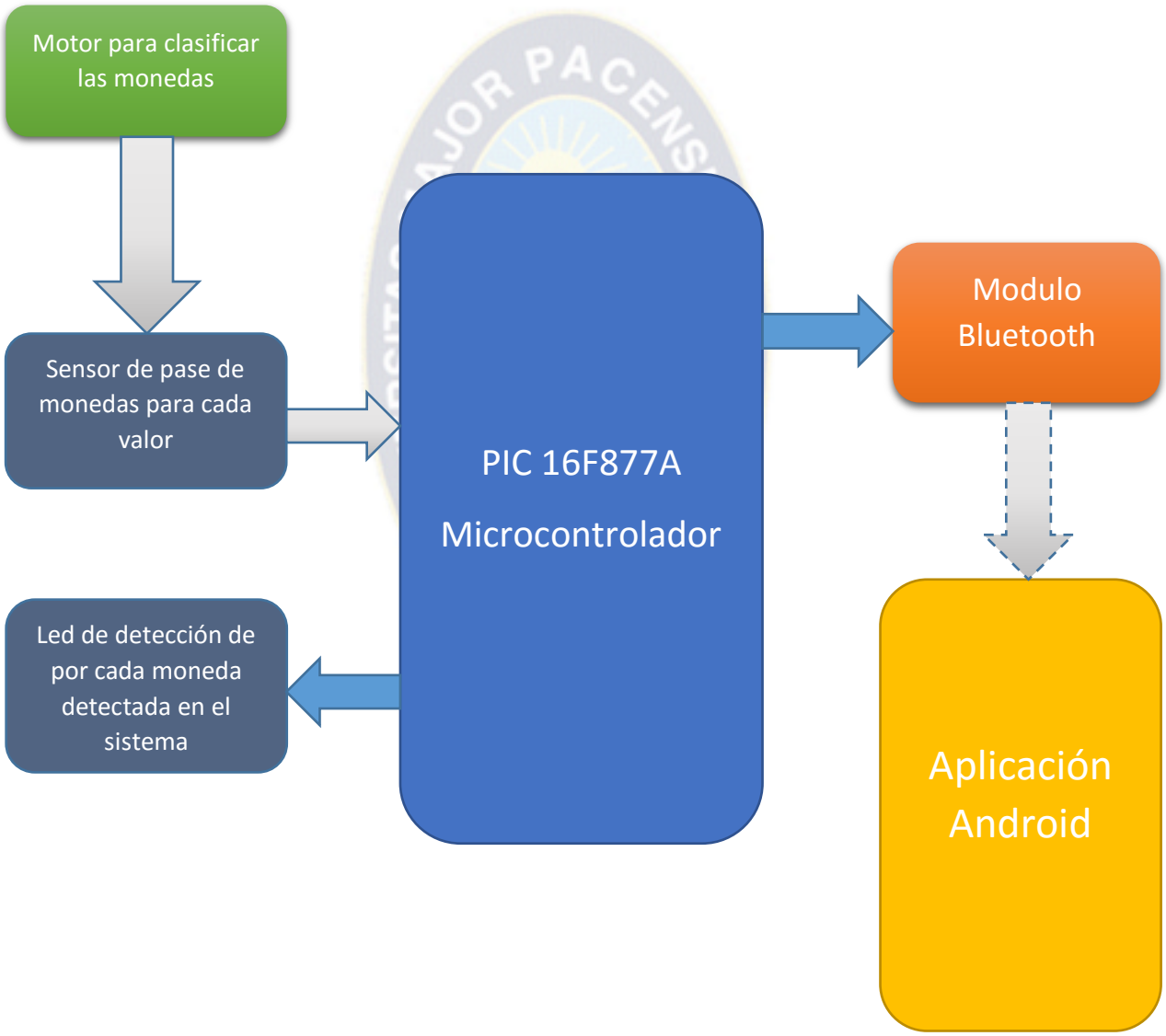


Figura Nº2 Diagrama de bloques del sistema de conteo
Fuente: Propia

3.3. Diagrama del circuito

Realiza un mecanismo que separe los valores de las monedas de acuerdo a su volar monetario(10 ctvs, 20ctvs, 50ctvs, 1Bs, 2Bs y 5Bs). Para realizar el conteo de cada tipo de moneda que ingrese al sistema. El siguiente circuito realiza esta operación:

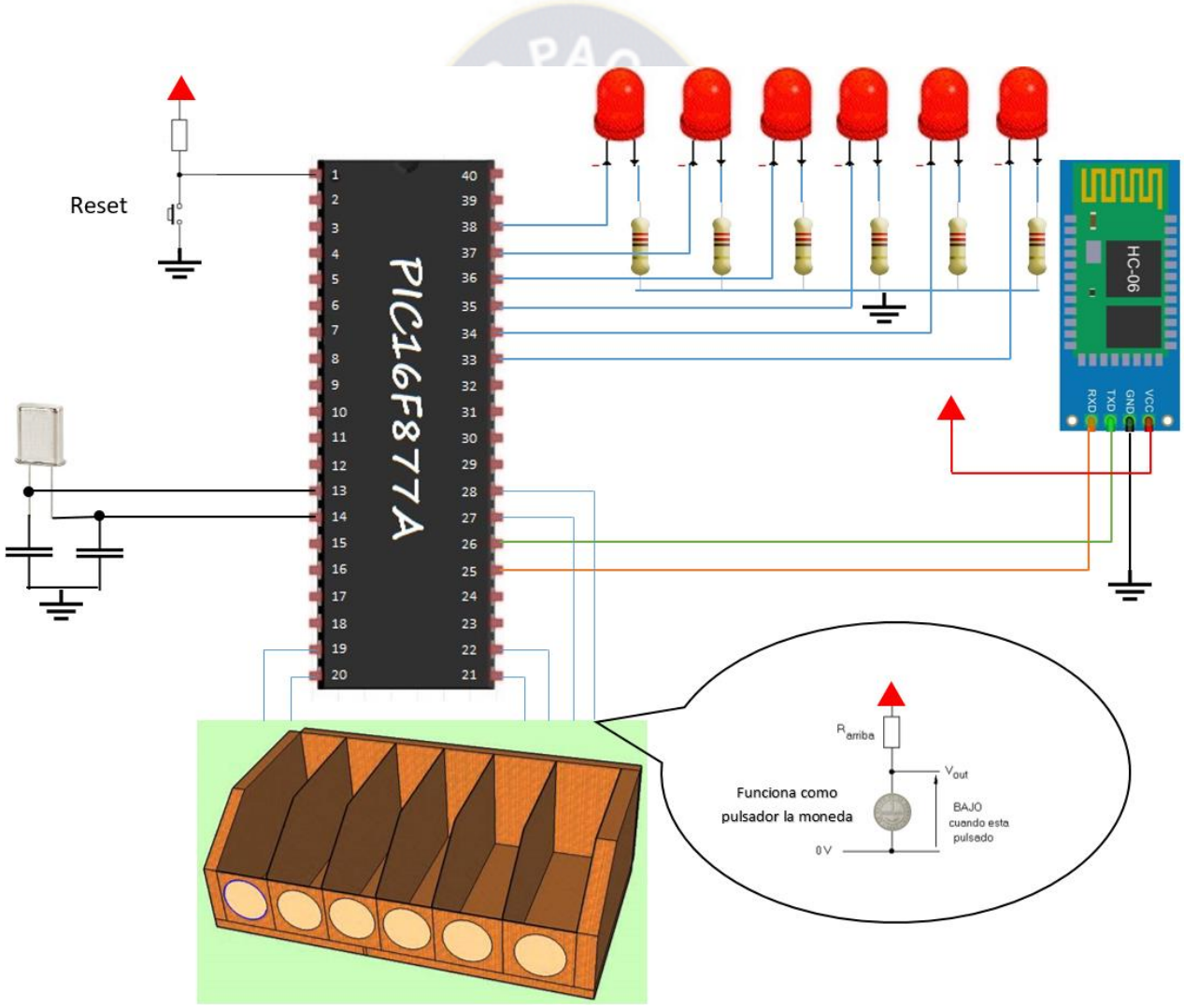


Figura Nº3 Diagrama del circuito del sistema de conteo
Fuente: Propia

3.4. Diagrama de la aplicación

Una vez que el PIC haya hecho la detección de cada moneda ingresada al sistema, este se mostrara en una aplicación Android llamada “Contador de Monedas” en donde se muestra las siguientes características:

Figura Nº4 Diagrama de la aplicacion del sistema de conteo
Fuente: Propia

3.5. Programación del contador en MikroBasic

Para hacer que el sistema de conteo funcione como se requiere, se realizó el siguiente codificacion en MikroBasic:

```
program contadorbasic
main:
TRISD=$FF /* configuramos el puerto D como entrada de datos*/
PORTB=$00 /* limpiamos el puerto B*/
TRISB=$00 /* configuramos el puerto B como salida de datos*/
UART1_Init(9600) /* inicializamos comunicador serial */
Delay_ms(100)
bucle:
if (PORTD.0=0) then /* detector de la moneda de 10 ctvs*/
  UART1_Write_Text("a") /* mandamos mensaje “a” hacia la aplicación android*/
  PORTB=$01
end if
if (PORTD.1=0) then /* detector de la moneda de 20 ctvs*/
  UART1_Write_Text("b") /* mandamos mensaje “b” hacia la aplicación android*/
  PORTB=$02
end if
if (PORTD.2=0) then /* detector de la moneda de 50 ctvs*/
  UART1_Write_Text("c") /* mandamos mensaje “c” hacia la aplicación android*/
  PORTB=$04
end if
if (PORTD.3=0) then /* detector de la moneda de 1Bs*/
```

```
  UART1_Write_Text("d") /* mandamos mensaje “d” hacia la aplicación android*/
  PORTB=$08
end if
if (PORTD.4=0) then /* detector de la moneda de 2Bs*/
  UART1_Write_Text("e") /* mandamos mensaje “e” hacia la aplicación android*/
  PORTB=$10
end if
if (PORTD.5=0) then /* detector de la moneda de 5Bs*/
  UART1_Write_Text("f") /* mandamos mensaje “f” hacia la aplicación android*/
  PORTB=$20
end if
delay_ms(100) /*tiempo aproximado que pasa una moneda por el sensor */
goto bucle
end.
```

3.6. Programación del contador en AppInventor2

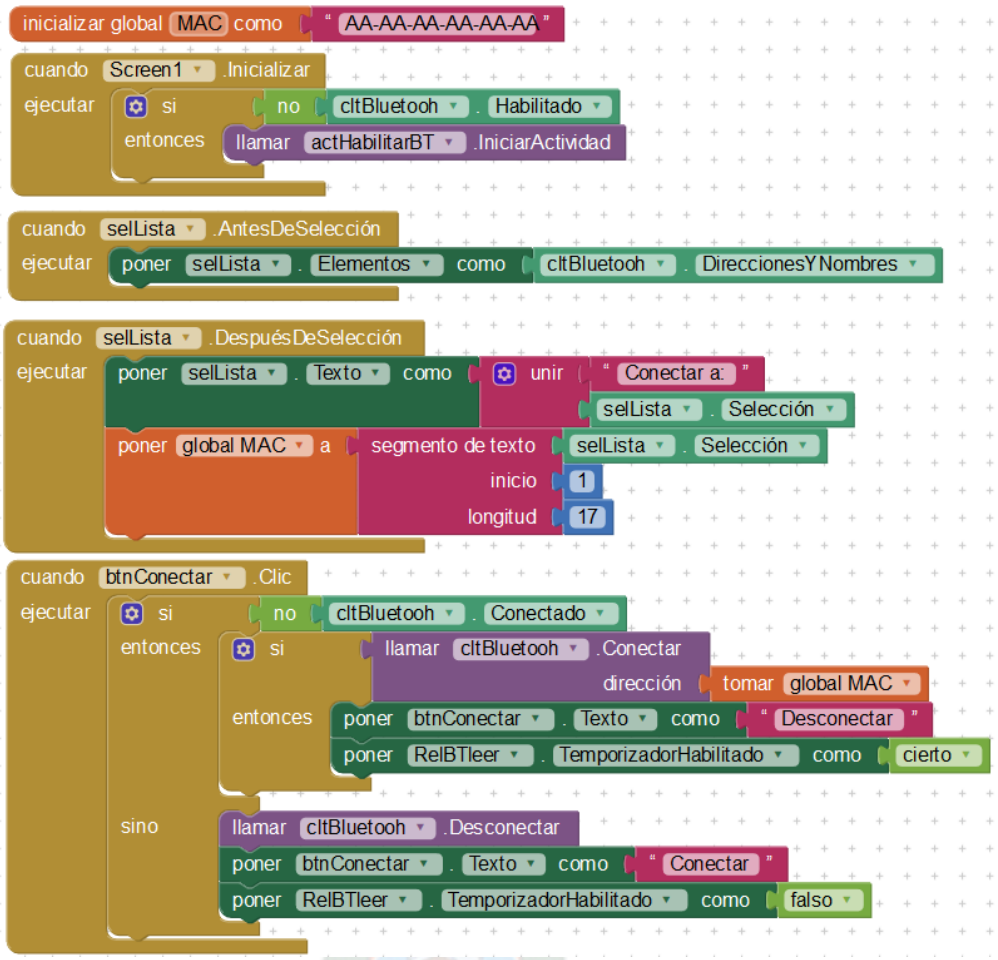
App Inventor puede realizar una inmensidad de funciones aplicativas en sistema operación android como ser conectividad bluetooth, diseño de gráficos, temporizador, actividades fuera del aplicación, etc. Estas fueron herramientas para que nuestra aplicación pueda comunicarse y realizar la función del conteo monetario.

El lenguaje de programación de basa en armado de bloques donde es muy didáctico ya que se arma en forma de rompencabezas,

El siguiente armado realizara las siguientes funciones para ejecutar la aplicación:

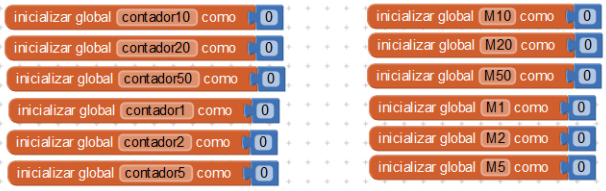
3.6.1. Conexión y desconexión con el modulo bluetooth HC-05

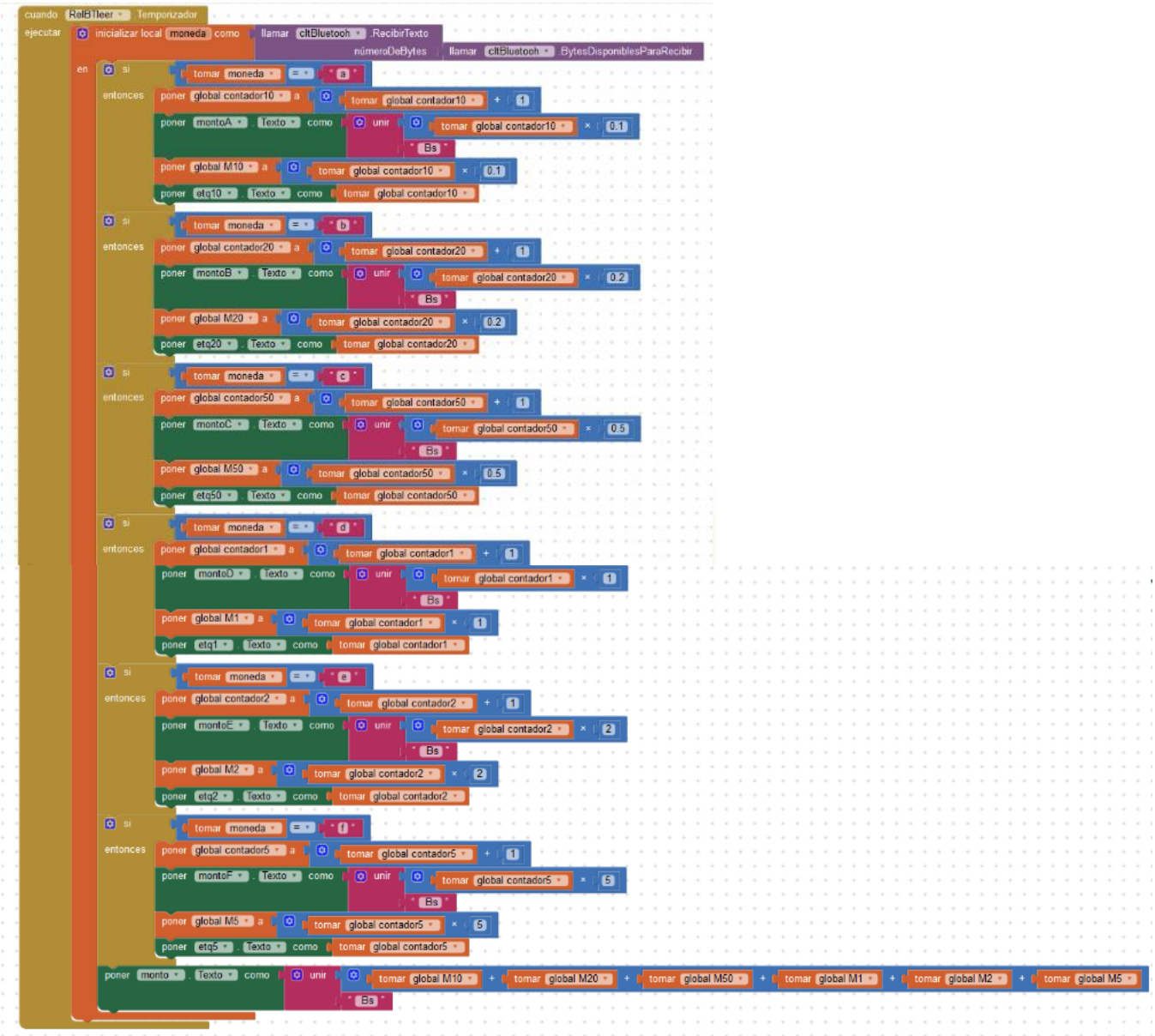
En esta parte realizamos la conexión y desconexión con nuestro modulo bluetooth:



3.6.2. Contador de monedas

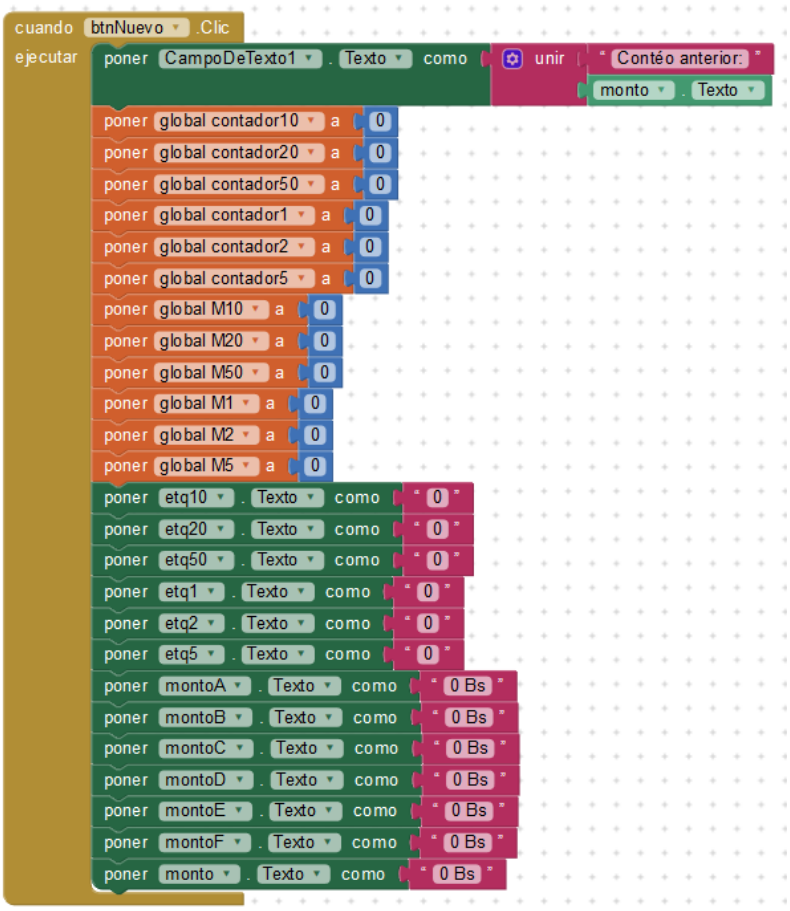
Una vez conectado recibiremos mensajes de nuestro PIC para ver que monedas se detectó en el sistema de conteo, para así poder realizar la suma de dinero total que el sistema conto para su control.





3.6.3. Nuevo Conteo

Para realizar un nuevo conteo de monedas al hacer clic en el botón “Nuevo conteo” este pondrá los valores predeterminados de la aplicación almacenando el último monto total en un campo de texto.



Costos:

Los costos que se requiere para realizar el circuito son:

Placa virgen	8 bs
Microcontrolador PIC16F877A	48 bs
Modulo bluetooth HC-05	80 bs
Oscilador de cristal	5 bs
2 capacitores de 22pF	2 bs
7 resistores de 10k ohm	1.4 bs
6 resistores de 220 ohm	1.2 bs
6 diodos led	6 bs
1 pulsador	2 bs

Esto hace una suma total de: 153.6 Bs

CAPÍTULO IV

4. CONCLUSIONES

Vemos que la clasificación y el conteo del total de monedas son precisa y confiable, verificando el control de cantidad de monedas exactas de cada valor monetario y el corto tiempo en que realiza en hacer la suma monetaria. Satisfaciendo el trabajo realizado de los operadores del servicio de transporte para su seguridad y confiabilidad en el momento de hacer cuentas.

Tambien:

- Se realizó el diseño de un hardware donde se realizamos el conteo de monedas mediante un mecanismo de separación de monedas de diferentes tipos de valores.
- Logramos realizar el software en MicroBasic para poder detectar el valor de cada moneda.
- Se logró configurar la comunicación serial gracias a una de sus librerías llamada Uart
- Logramos realizar el software en App Inventor para un teléfono Smartphone donde se puede observar que cantidad de dinero ingreso al sistema.
- Pudimos sincronizar la frecuencia de trabajo para que nuestro modulo bluetooth se comunique con nuestro celular.
- Se obtuvieron la cifra decimal que indica la cantidad de dinero que nuestro sistema detecto al pasar por los sensores de conteo.

CAPÍTULO V

5. BIBLIOGRAFIA

1 Hector Morlote (2012-2013) Estudio de la estructura interna del PIC16F877A de https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-del-pic-16f877/

2 Laura Sofia Avellaneda - Brandon Cortes Caicedo (24 noviembre 2017) Contador de monedas de https://digitales573.wordpress.com/2017/11/24/contador-de-monedas/

3 Carlos González Morcillo y Sergio García Mondaray. (Diciembre de 2011) Curso de App Inventor de http://webpub.esi.uclm.es/img/upload/plugin/ESI-TechLab-AppInventor2-2015beta.pdf

4 Geek Factory (21 febrero del 2014) Geek Factory Bluetooth HC-05 y HC-06 Tutorial de Configuración de https://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/

5 MIT APP INVENTOR (July 31 2018 Versión: nb170) http://ai2.appinventor.mit.edu/?locale=es_ES#5272326071517184

6 MrElberni (2015) USART PIC Comunicación serial de http://microcontroladores-mrelberni.com/usart-pic-comunicacion-serial/

ANEXOS

PIC16F877A Pin Configuration

Pin Number	Pin Name	Description
1	MCLR/Vpp	MCLR is used during programming, mostly **connected to programmer** like PicKit
2	RA0/AN0	**Analog pin 0** or 0th pin of PORTA
3	RA1/AN1	**Analog pin 1** or 1st pin of PORTA
4	RA2/AN2/Vref-	**Analog pin 2** or 2nd pin of PORTA
5	RA3/AN3/Vref+	**Analog pin 3** or 3rd pin of PORTA
6	RA4/T0CKI/C1out	4th pin of PORTA
7	RA5/AN4/SS/C2out	**Analog pin 4** or 5th pin of PORTA
8	RE0/RD/AN5	**Analog pin 5** or 0th pin of PORTE
9	RE1/WR/AN6	**Analog pin 6** or 1st pin of PORTE
10	RE2/CS/AN7	7th pin of PORTE
11	Vdd	**Ground pin** of MCU
12	Vss	**Positive pin** of MCU (+5V)
13	OSC1/CLKI	**External Oscillator**/clock input pin
14	OSC2/CLKO	**External Oscillator**/clock output pin
15	RC0/T1OSO/T1CKI	0th pin of PORT C
16	RC1/T1OSI/CCP2	1st pin of POCTC or **Timer/PWM** pin
17	RC2/CCP1	2nd pin of POCTC or **Timer/PWM** pin
18	RC3/SCK/SCL	3rd pin of POCTC
19	RD0/PSP0	0th pin of POCTD
20	RD1/PSPI	1st pin of POCTD
21	RD2/PSP2	2nd pin of POCTD
22	RD3/PSP3	3rd pin of POCTD
23	RC4/SDI/SDA	4th pin of POCTC or Serial Data in pin
24	RC5/SDO	5th pin of POCTC or Serial Data Out pin
25	RC6/Tx/CK	6th pin of POCTC or **Transmitter pin** of Microcontroller
26	RC7/Rx/DT	7th pin of POCTC or **Receiver pin** of Microcontroller
27	RD4/PSP4	4th pin of POCTD
28	RD5/PSP5	5th pin of POCTD
29	RD6/PSP6	6th pin of POCTD
30	RD7/PSP7	7th pin of POCTD
31	Vss	**Positive pin** of MCU (+5V)
32	Vdd	**Ground pin** of MCU

33	RB0/INT	0th pin of POCTB or **External Interrupt** pin
34	RB1	1st pin of POCTB
35	RB2	2nd pin of POCTB
36	RB3/PGM	3rd pin of POCTB or **connected to programmer**
37	RB4	4th pin of POCTB
38	RB5	5th pin of POCTB
39	RB6/PGC	6th pin of POCTB or **connected to programmer**
40	RB7/PGD	7th pin of POCTB or **connected to programmer**

PIC16F877A Features

PIC16F877A –Simplified Features	
CPU	8-bit PIC
Number of Pins	40
Operating Voltage (V)	2 to 5.5 V
Number of I/O pins	33
ADC Module	8ch, 10-bit
Timer Module	8-bit(2), 16-bit(1)
Comparators	2
DAC Module	Nil
Communication Peripherals	UART(1), SPI(1), I2C(1), MSSP(SPI/I2C)
External Oscillator	Up to 20Mhz
Internal Oscillator	Nil
Program Memory Type	Flash
Program Memory (KB)	14KB
CPU Speed (MIPS)	5 MIPS
RAM Bytes	368
Data EEPROM	256 bytes