UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA

MATHEUS ANTUNES CHIPANSKI
MILAN AVILA CLASEN

PLATAFORMA DE PROGRAMAÇÃO PARA MICROCONTROLADOR PIC ATRAVÉS DE APLICATIVO ANDROID

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2016

MATHEUS ANTUNES CHIPANSKI
MILAN AVILA CLASEN

PLATAFORMA DE PROGRAMAÇÃO PARA MICROCONTROLADOR PIC ATRAVÉS DE APLICATIVO ANDROID

Trabalho de Conclusão de Curso apresentado na disciplina de Trabalho de Conclusão de Curso 2 do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Engenheiro de Controle e Automação"

Orientador: Prof. Dr. Glauber Gomes de Oliveira Brante

CURITIBA
2016

Matheus Antunes Chipanski
Milan Avila Clasen

PLATAFORMA DE PROGRAMAÇÃO PARA MICROCONTROLADOR PIC ATRAVÉS DE APLICATIVO ANDROID

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro de Controle e Automação, do curso de Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 13 de Dezembro de 2016.

_____
Prof. Paulo Sérgio Walenia, Esp.
Coordenador de Curso
Engenharia de Controle e Automação

_____
Profa. Annemarlen Gehrke Castagna, Ma.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia de Controle e Automação do DAELT

ORIENTAÇÃO

_____
Prof. Dr. Glauber Gomes de Oliveira Brante
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Prof. Dr. Glauber Gomes de Oliveira Brante
Universidade Tecnológica Federal do Paraná

_____
Prof. Dr. Marco Jose da Silva
Universidade Tecnológica Federal do Paraná

_____
Prof. Dr. Guilherme Moritz
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenaria de Controle e Automação

AGRADECIMENTOS

Ao Professor Glauber Brante, pela orientação e o apoio;

Aos Professores Guilherme Moritz e Marco da Silva, pelas sugestões e correções;

A Lucas Lugnani Fernandes, por muitos auxílios;

A Thyane e Camila, pelo apoio e carinho;

A Masahiro Sakurai, por ter unido os membros da equipe;

Nossos mais profundos agradecimentos.

RESUMO

CHIPANSKI, Matheus Antunes; CLASEN, Milan Avila. PLATAFORMA DE PROGRAMAÇÃO PARA MICROCONTROLADOR PIC ATRAVÉS DE APLICATIVO ANDROID. 44 f. Trabalho de conclusão de curso – Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

Neste trabalho são apresentados os materiais, ferramentas e procedimentos utilizados para o desenvolvimento de uma plataforma de programação de microcontroladores através de um aplicativo para dispositivos *mobile*. O trabalho se divide em uma aplicativo para celular e uma plataforma microcontrolada, onde a plataforma microcontrolada opera com um *firmware* próprio, adequado a receber dados do aplicativo através de uma sequencia de comunicação propria que utiliza comunicação *Bluetooth*. A plataforma microcontrolada desenvolvida está apta para operar a partir das informações contidas em um fluxograma de operação criado pelo usuário na interface gráfica do aplicativo, onde cada bloco do fluxograma corresponde a uma funcionalidade da plataforma microcontrolada pré-programada no microcontrolado. O trabalho objetiva fornecer um ambiente de programação acessível e compreensível, abstraindo a maior parte das dificuldades existentes quando se esta estabelecendo os componentes básicos de uma aplicação envolvendo microcontroladores.

Palavras-chave: Android, PIC, Bluetooth, Desenvolvimento, Microcontrolador

ABSTRACT

CHIPANSKI, Matheus Antunes; CLASEN, Milan Avila. PIC Microcontroller programming platform through android application. 44 f. Trabalho de conclusão de curso – Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

In this paper we present the equipment, tools and procedures used in the development of a programming platform for microcontrollers through the use of a mobile app. The paper is divided in a mobile app and a microcontrolled platform operating on a in-house firmware, prepared to receive data from the app through a particular sequence of Bluetooth messages also described in the paper. The microcontrolled platform developed is apt to operate based on the information contained in an operation flowchart created by the user through the use of the app's graphical user interface, where each block in the flowchart corresponds to a specific functionality of microcontrolled platform pre-programmed in the microcontroller. This paper has as objective to establish a programming environment that is both accessible and easy to understand, leaving most of the set-up difficulties associated with microcontroller programming up to the platform.

Keywords: Android, PIC, Bluetooth, Development, Microcontroller

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A programação dos microcontroladores permite sua utilização em várias áreas, mas apresenta uma barreira considerável para os usuários sem experiência prévia com programação, ou montagem de sistemas digitais/eletrônicos. Com o intuito de facilitar o acesso a essas tecnologias surgiram no mercado alguns *kits* microcontroladores, como o *Arduíno* (MCROBERTS, 2011). O *Arduíno* é uma plataforma de prototipagem de código aberto baseada em *hardware* e *software* de fácil utilização e que facilita o acesso e manipulação das suas entradas e saídas, propondo uma linguagem de programação própria baseada em linguagem C, mas com diversas *APIs* (do inglês, *Application Programming Interface*) automatizadas para lidar com periféricos variados. As *APIs* do *Arduíno* são de fácil compreensão e atribuem ao usuário grande facilidade de uso. Desde seu surgimento a plataforma teve grande participação na popularização de pequenos sistemas eletro-eletrônicos e é utilizada com frequência entre usuários amadores e iniciantes.

Além disso, o grande avanço e a popularização dos dispositivos móveis tem promovido grande inclusão digital e despertado interesse por aplicativos e tecnologias portáteis. Ademais, atualmente pode-se facilmente encontrar aplicações diversas que utilizam dispositivos móveis, tais como *tablets* e *smartphones*, para aplicações profissionais, como o monitoramento plantas produtivas (LIAN et al., 2013). Um dos sistemas operacionais mais comuns no mercado hoje em dia é o sistema *Android* da *Open Handset Alliance*, uma aliança entre várias empresas, dentre elas a *Google*. Este sistema operacional baseado em *Linux* é bastante adaptável e tem forte suporte à criação de *sotwares* de desenvolvedores que não estejam ligados diretamente à *Open Handset Alliance*, como é o caso deste projeto.

Assim, tendo em vista que existem muitos cenários nos quais a integração entre a automação microcontrolada e o controle operacional remoto se fazem necessários, este projeto visa desenvolver uma plataforma que auxilie na ligação entre essas duas tecnologias. Dentre as várias tecnologias de comunicação sem fio disponíveis nos dispositivos *Android* se destaca o *Bluetooth*. *Bluetooth* é uma especificação para a comunicação em curta distância de redes sem fio com um baixo custo (KOBAYASHI, 2004). Sua comunicação de curto alcance permite a troca de dados por meio de um sinal de rádio dentro da faixa de 2,4 a 2,485 GHz, com alcance de cerca de 10 metros para dispositivos *Bluetooth* 4.0 de classe 2 (BLUETOOTH, 2010).

Mesmo frente aos grandes avanços na área, a criação e programação de sistemas microcontrolados ainda é um processo relativamente complexo e pouco atraente para leigos ou iniciantes. O número de parâmetros e detalhes a serem considerados, até mesmo antes do início da criação dos sistemas propriamente ditos, já apresenta uma grande barreira aos possíveis entusiastas ou profissionais futuros. Problemas tais como a conexão do controlador do sistema com o ambiente de programação, a configuração de *drivers* e até mesmo o grande detalhismo exigido pela sintaxe de muitas das linguagens de programação são muitas vezes problemas que desmotivam o desenvolvedor em potencial ou inviabilizam protótipos mais simples.

1.1 OBJETIVOS

O projeto tem como objetivo a criação de um sistema microcontrolado para que possua interface com um aplicativo para *Android* também a ser desenvolvido. Este permitirá ao usuário controlar as entradas e saídas do microcontrolador e programar alguns processos simples através de uma interface gráfica. Os requisitos mínimos do sistema estão descritos na Seção 2.1.

1.1.1 Objetivos Específicos

- Desenvolver um *software Android* que atue como um ambiente de programação com interface gráfica de simples compreensão e possua a capacidade de comunicar-se via *Bluetooth* a um sistema microcontrolado, enviando comandos para reprogramá-lo para diferentes tarefas;
- Criar um sistema microcontrolado capaz de receber comandos via comunicação *Bluetooth* e reconfigurar suas entradas e saídas de acordo com o sinal recebido. Ao mesmo tempo disponibilizando-as para uso do desenvolvedor;
- Implementar um sistema de comunicação via *Bluetooth* entre o sistema microcontrolado e o aplicativo da plataforma *Android*;
- Desenvolver como projeto exemplo um pequeno carro controlado eletronicamente, cujo comportamento possa ser manipulado através de um microcontrolador acoplado e o aplicativo *Android*.

1.1.2 Aplicações

O projeto da plataforma de programação objetivada neste trabalho busca criar uma maneira fácil e rápida de programar microcontroladores por meio de dispositivos móveis. Essa forma de programação abre possibilidades para usuários de todos os níveis de conhecimento uma vez que simplifica o trabalho para os mais inexperientes, agiliza o trabalho daqueles que já estão familiarizados e também possibilita a programação e desenvolvimento de protótipos em campo, seja ele qualquer ambiente, sem estar preso a um computador ou laboratório.

Uma vez que o usuário tem acesso ao processador do dispositivo móvel e os periféricos do microcontrolador preparados para comandar as funções que forem necessárias, as possibilidades de criação e experimentação são as mais diversas. Protótipos de carrinhos, braços robóticos simples, sistemas de automação caseiros, sistemas de iluminação, brinquedos, entre outros. Outro objetivo da interface é apresentar os básicos da programação do microcontrolador, podendo tornar a plataforma desenvolvida uma ferramenta de ensino de grandes possibilidades e uma curva de aprendizado muito menor.

Outras aplicações são imaginadas, uma vez que novas funções que não estão no escopo deste trabalho já são parte de planos futuros ao desenvolvimento deste projeto. Algumas destas projeções serão discutidas posteriormente no Capítulo 6.

1.2 ESTRUTURA DO TRABALHO

A sequência desse trabalho está organizada da seguinte maneira:

- Capítulo 2 - Especificações técnicas: Esse capítulo objetiva detalhar as abordagens adotadas para o desenvolvimento das partes do projeto, analisando separadamente o desenvolvimento da plataforma microcontrolada, o aplicativo *Android* e o protocolo de comunicação escolhido. Está no escopo desse capítulo justificar as escolhas de ferramentas para o desenvolvimento do *software* e para a montagem da plataforma de testes, além de definir como será o fluxo de funcionamento do processo como um todo.
- Capítulo 3 - Operação: Esse capítulo detalha como o usuário cria o próprio programa a ser executado pela plataforma, incluindo as fases de preparação e programação do projeto.
- Capítulo 4 - Funções disponíveis: uma listagem das funções a serem utilizadas pelo usuário junto à explicação do funcionamento de cada uma quanto à configuração e efeitos na plataforma microcontrolada e aplicativo.

- Capítulo 5 - Aplicações Exemplo: Aqui serão apresentadas dois exemplos de aplicações montados para a defesa do projeto utilizando os sistemas desenvolvidos.
- Capítulo 6 - Conclusão: Neste capítulo é dado encerramento ao trabalho, com apresentação e discussão dos resultados obtidos, e algumas perspectivas futuras para o aplicativo.

2 ESPECIFICAÇÕES TÉCNICAS

Neste capítulo serão apresentados os requisitos mínimos estabelecidos no começo do desenvolvimento do trabalho, seguido das ferramentas necessárias para desenvolvimento do projeto, bem como as especificações de *hardware* e *software* a serem utilizadas. Além disso, também serão apresentadas as escolhas e justificativas para equipamentos, *software* e protocolo de comunicação. No final deste capítulo será apresentado o fluxo de funcionamento do projeto, explicitando cada etapa envolvida.

A Seção 2.1 fornece uma visão geral do processo, a Seção 2.2 vai definir quais são os requisitos mínimos do projeto (funcionais e não-funcionais). A seguir, a Seção 2.3 vai detalhar a plataforma microcontrolada, a Seção 2.4 vai detalhar o aplicativo *Android* e a Seção 2.5 vai detalhar o protocolo de comunicação *Bluetooth* e como ele será implementado. Por fim, na Seção 2.6 temos um fluxograma ilustrando o funcionamento do processo como um todo.

2.1 VISÃO GERAL

O projeto, na sua forma geral, pode ser ilustrado na Figura 1. Existem duas plataformas principais a serem desenvolvidas, a plataforma *Android* e a plataforma microcontrolada, sendo a comunicação entre elas realizada através de uma interface *Bluetooth*.

Primeiramente, a plataforma *Android* será responsável por receber comandos do usuário de maneira gráfica e intuitiva, para então transformar esses comandos em uma estrutura de dados padronizada. Por sua vez, a plataforma microcontrolada deve ser capaz de receber essa estrutura de dados e interpretar os comandos do usuário, a seguir entrando em um laço de operação que executará as instruções do usuário sequencialmente. Por fim, a comunicação *Bluetooth* está encarregada de permitir a transmissão de dados entre o aplicativo *Android* e a plataforma microcontrolada.

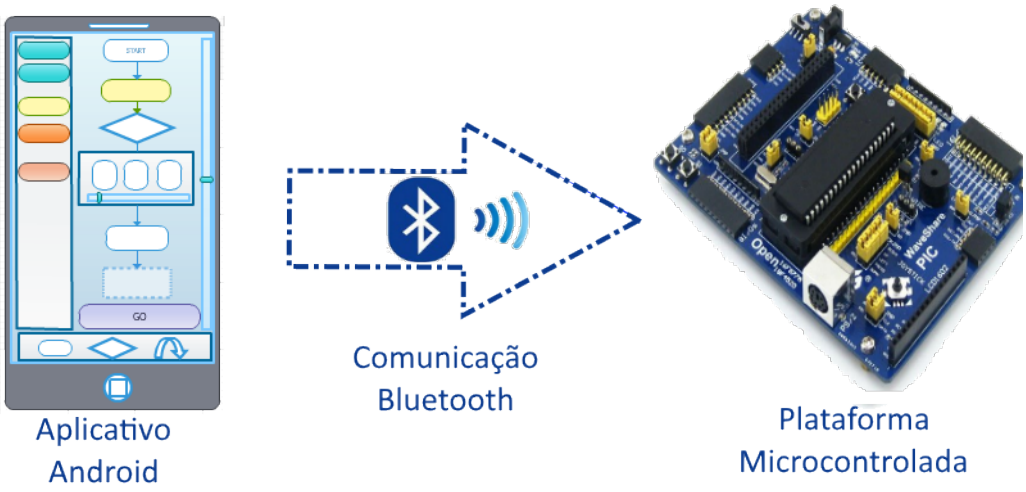

Figura 1: Diagrama Simplificado do Projeto.

Fonte: Autoria própria.

2.2 REQUISITOS MÍNIMOS DO SISTEMA

Nesta seção apresentaremos os requisitos mínimos separados em requisitos funcionais e requisitos não-funcionais para ambas as partes do projeto: o aplicativo *Android* e a plataforma microcontrolada.

- Aplicativo *Android* - Requisitos Funcionais:

  1. O aplicativo deve permitir que o usuário construa um fluxograma de operação, constituído de uma série de funções pré-definidas de acordo com a especificação do Capítulo 4 deste documento.
  2. O aplicativo deve permitir que o usuário inicie a conexão via comunicação *Bluetooth* entre o dispositivo *Android* e o módulo HC-05 (REUTLINGEN-UNIVERSITY, 2016) da plataforma microcontrolada.
  3. Uma vez estabelecida a conexão entre o dispositivo *Android* e a plataforma microcontrolada o aplicativo deve permitir que o usuário inicie o envio do fluxograma de operação para a plataforma microcontrolada.
  4. A interface do aplicativo deve permitir que o usuário selecione as funções a serem adicionadas ao fluxograma de operação a partir de uma lista formada pelas funções especificadas no Capítulo 4 deste documento.
  5. A interface do aplicativo deve conter uma representação gráfica do fluxograma montado pelo usuário.

6. As funções que compõem os blocos do fluxograma devem conter os valores necessários para a sua execução (e.g., A função *Port Out* deve conter a porta a qual se dirige e o valor que ela deve assumir). Estes valores devem ser alteráveis pelo usuário por meio da interface gráfica.

- Aplicativo *Android* - Requisitos Não-Funcionais:

1. O aplicativo deve ser desenvolvido na linguagem JAVA, com a IDE Android Studio, pelos motivos descritos na Seção 2.3 deste documento.
2. As funções especificadas no Capítulo 4 deste documento devem ser traduzidas para o aplicativo na forma de um objeto do tipo *Function*, que contenha os valores necessários específicos para cada uma das funções descritas.
3. Uma lista contendo todas as funções especificadas no Capítulo 4 deste documento deve servir como base para a criação de objetos do tipo *FluxogramBlock*, que contém um objeto do tipo *Function* e outros parâmetros relacionados a representação gráfica do bloco no fluxograma.
4. Os objetos do tipo *FluxogramBlock* devem ser organizados no formato de um grafo, de modo que cada nó deste grafo contenha um *FluxogramBlock* e aponte para pelo menos um outro.
5. A comunicação *Bluetooth* deve ser gerenciada por uma *Thread* separada da principal, de modo a não bloquear atualizações dos elementos de interface. As *Threads* de comunicação e a principal devem se comunicar através de um objeto do tipo *Control*, instanciado na *Thread* principal, mas referenciado por ambas.
6. Os valores associados a cada função devem ser separados e convertidos em *arrays* de no máximo três *bytes* (limite do *buffer* do *PIC16F877A* (MICROCHIP, 2013)).
7. Em casos onde a função necessitar de mais de três *bytes* para ser enviada o aplicativo deve executar uma rotina de envio capaz de realizar múltiplos envios para o microcontrolador.
8. Uma vez iniciado o envio do fluxograma de operação a *Thread* de comunicação deve ser capaz de percorrer o grafo formado pelos objetos do tipo *FluxogramBlock*, enviando os cada bloco individualmente, seguindo a sequência indicada pelo grafo. A *Thread* de comunicação deve enviar um *array* de *bytes* para cada *FluxogramBlock* e aguardar o recebimento de um comando da plataforma microcontrolada para enviar o seguinte.

- Plataforma Microcontrolada - Requisitos Funcionais:

1. A plataforma deve ser capaz de receber os comandos do aplicativo e executá-los de acordo com o planejado pelo usuário.
2. Deve ser possível interromper a execução da sequência no próprio microcontrolador, caso seja necessário.
3. É necessário que a conexão *Bluetooth* esteja sempre visível ao usuário dado que a plataforma esteja ligada e pronta para operar.
4. A plataforma deve ser capaz de controlar diferentes atuadores e sensores que possam ser conectados ao microcontrolador, de acordo com as funções descritas no Capítulo 4 deste documento.
5. Caso qualquer informação enviada não possa ser processada, o usuário deverá ser informado.

- Plataforma Microcontrolada - Requisitos Não-Funcionais:

1. A plataforma deve ser capaz de interpretar a informação enviada pelo dispositivo *Android* de forma estruturada para controlar os periféricos do microcontrolador de forma a ser fiel à tarefa requisitada pelo usuário.
2. É necessária uma proteção mínima para que o *buffer* do *Bluetooth* não seja sobrecarregado.

2.3 PLATAFORMA MICROCONTROLADA

O protótipo desenvolvido foi criado com o *PIC16F877A* da *Microchip* e se utiliza do *kit Micropic - DAELT* desenvolvido e utilizado na Universidade Tecnológica Federal do Paraná. A primeira versão da plataforma visa a implementação do fluxo de dados para que mais tarde seja possível criar uma plataforma mais abrangente em um formato adaptável, compatível também com um maior número microcontroladores.

2.3.1 Micropic DAELT

A plataforma de desenvolvimento criada e utilizada na instituição de ensino a qual a equipe envolvida neste projeto trabalhou, o *Kit Micropic* mostrado na foto da Figura 2, foi escolhida como a base deste projeto. Tal plataforma oferece grande número de circuitos para fins didáticos, possibilitando assim acesso facilitado a, por exemplo, comunicação serial

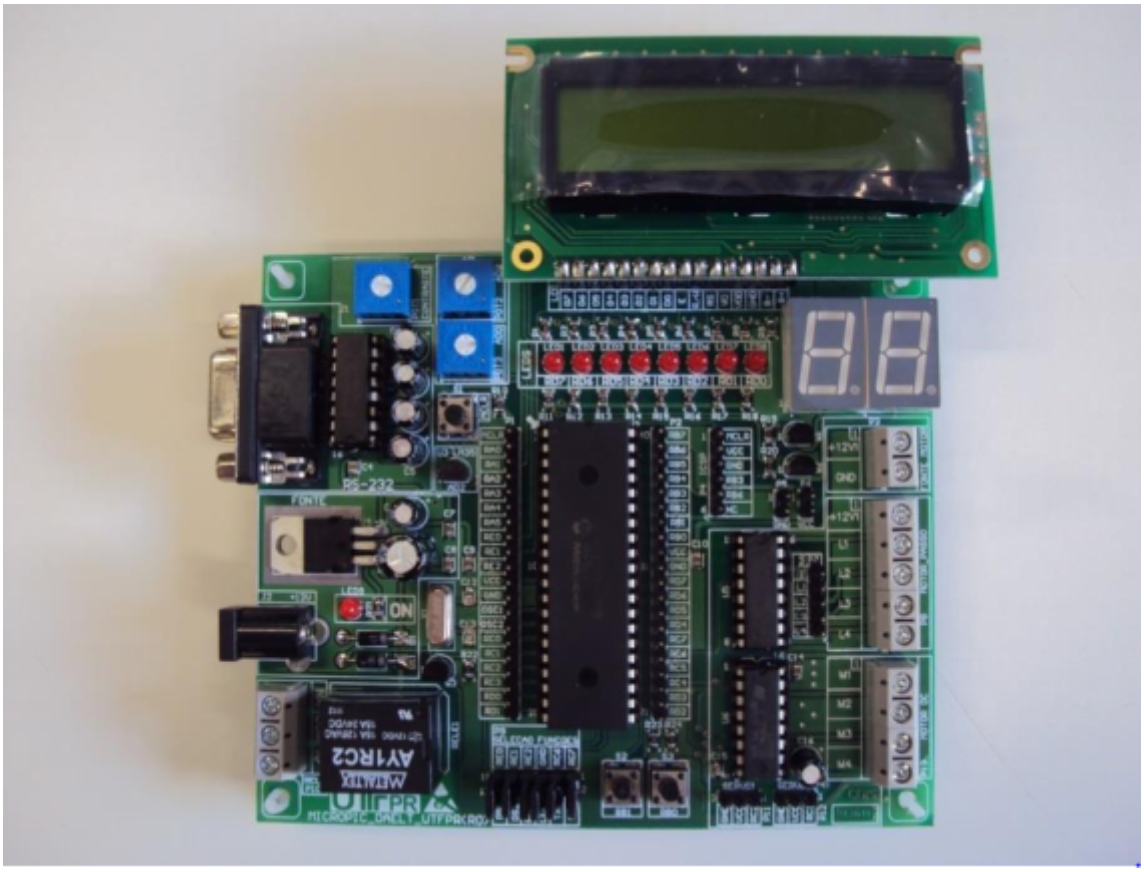

Figura 2: Sistema embarcado MICROPIC - DAELT.
Fonte: Manual MICROPIC - DAELT.

e módulos de comando de motores de passo, o que elimina a necessidade de aquisição de periféricos externos, comumente conhecidos como *shields*.

O *Kit Micropic* possibilita controle de motores de passo e motores de corrente contínua através dos periféricos *ULN2003A* (JONES, 2007) e L293D (INSTRUMENTS, 2010) já instalados, para aplicações de automação além do relé NA/NF que pode servir para acionamento de circuitos de maior potência. Também podemos encontrar nele o circuito controle para utilização de um LCD alfanumérico 16x2 no modo 4 e 8 *bits*, dois *displays* de sete segmentos multiplexados e uma porta de comunicação serial padrão *RS-232* fêmea com modulador *MAX232* (INSTRUMENTS, 2004), úteis para as mais diferentes indicações e interfaces com o usuário. Uma vez que o projeto se dedica ao acesso e manipulação das diferentes funcionalidades de um microcontrolador, o controle de todos estes periféricos pode demonstrar a aplicabilidade do projeto em diferentes projetos como, por exemplo, o controle de um carrinho guiado por uma sequência de comandos controlando motores e com diferentes indicadores.

O microcontrolador utilizado para a confecção do *Kit Micropic* foi o *PIC16F877A* da *Microchip*. Os principais fatores que influenciaram esta decisão foram o custo reduzido frente ao número de funcionalidades e a acessibilidade ao mesmo graças à sua popularidade no mercado micro-eletrônico de Curitiba.

De acordo com a *datasheet* do microcontrolador o *PIC16F877A*, como mostrado na Figura 3, possui trinta e três portas *I/O*, três *timers*, dois módulos de comparação/captura/*PWM*, uma porta paralela de oito *bits*, porta serial com capacidade para *SPI* (*Serial Peripheral Interface*) e *I2C*^{TM} (*Inter-Integrated Circuit*), oito canais digital/analógico, uma interface *UART* (*Universal asynchronous receiver/transmitter*) e opera em uma faixa de 2V a 5,5V. Quando se considera todos os periféricos e o investimento necessários, concluiu-se que este modelo tem boa relação custo benefício para as aplicações mais comuns da plataforma que desejávamos criar, como o *Kit Micropic* possui todos os circuitos necessário para a sua utilização este é o ponto de partida mais adequado para o projeto.

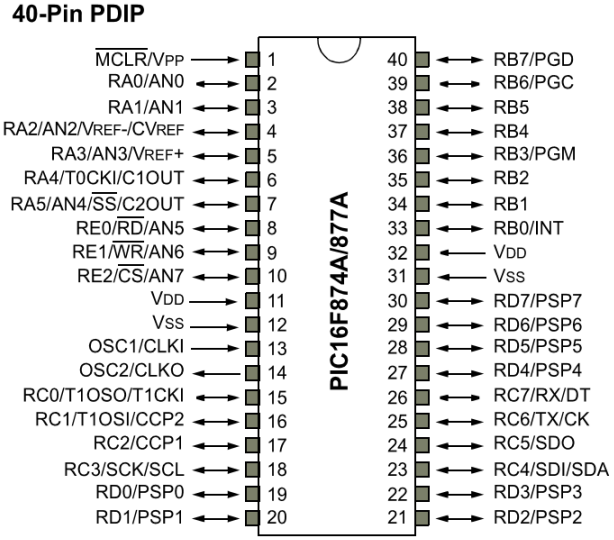

Figura 3: Diagrama do PIC1F877A.

Fonte: ***Datasheet PIC16F8877A - Microchip.***

2.3.2 *Firmware*: Programação em linguagem C

Para facilitar o trabalho de gravação do microcontrolador e cortar gastos com um circuito gravador, inicialmente utilizou-se um programa *bootloader* no sistema. *Bootloaders* são pequenos programas que, uma vez gravados na memória do microcontrolador, permitem que a a memória *ROM* seja alterada por meio de comunicação serial (GOODMAN, 2006). Então o

protótipo pode ser reprogramado de maneira muito mais simples e de forma independente pela equipe sem necessidade de realizar os testes apenas dentro dos laboratórios do campus.

O *firmware* utilizado neste projeto será desenvolvido e programado na *IDE MPLAB* versão 8.91 para *Windows* (MPLAB, 2006). O programa foi escolhido no processo de experimentação de diferentes *IDE*s (*Integrated Development Enviroment*) e selecionado por ser capaz de suportar o compilador *HI-TECH C* (HI-TECH, 2010) da *Microchip* e gerar um arquivo *.HEX* aceito pelo nosso *bootloader* escolhido, o *TinyMultiBootloader* (SOURCEFORGE, 2016), responsável pela atualização do *firmware* durante o desenvolvimento do projeto. Além disso, o ambiente de programação se provou eficiente e organizado. Além disso a IDE apresenta uma funcionalidade de *debugger* do código mais robusta, tornando as correções do programa simples o bastante para manter o trabalho no código em bom ritmo.

O compilador *HI-TECH C* permite a programação do microcontrolador em linguagem C, simplificando e acelerando o trabalho de construção do código. Ele possui uma variada biblioteca de funções específicas para a programação de microcontroladores *PIC* garantindo, por exemplo, acesso rápido aos registros por meio de palavras-chave, permitindo sua alteração em poucas linhas de programa e mantendo a objetividade e clareza do código.

Posteriormente no desenvolvimento do trabalho, o *Bootloader* não era mais confiável o suficiente para ser utilizado devido ao grande volume de dados aplicados no *firmware*. Sua utilização foi então substituída pela aplicação do gravador *PKBurner* (CARELLE, 2016). O *PKBurner* é um *debugger* para gravação de microcontroladores da *Microchip* por meio de *USB* e substituiu de maneira fácil a gravação por *Bootloader* uma vez que o *Micropic* já possui interface para a utilização do mesmo.

O funcionamento do *firmware* se resume na seleção da função desejada por meio do primeiro *Byte* recebido e execução da mesma aplicando as configurações competentes a medida que são recebidas pelos *Bytes* seguintes. Caso o Buffer da antena *Bluetooth* não tenha mais informação, o *firmware* pedirá para que o próximo pacote seja enviado pelo aplicativo. A visualização de seu funcionamento completo está demonstrada na Figura 4.

O *firmware* possui funções correspondentes às do aplicativo, elas são selecionadas por meio de uma interrupção de recebimento de sinal pela *UART* o qual entra em uma grande estrutura de *Switch Case* que busca, por meio da *Id* da função desejada, contida no primeiro *byte*, chamar a função correta, desejada pelo usuário. Uma vez que a função é selecionada, ela recebe os *bytes* seguintes do pacote e os direciona para as respectivas configurações de acordo com a função selecionada. Caso seja necessário, ao final do recebimento do pacote, algumas funções mandam uma requisição de novo pacote e assim por diante. Assim que tudo

está configurado a função é executada é enviado um sinal de conclusão para o aplicativo.

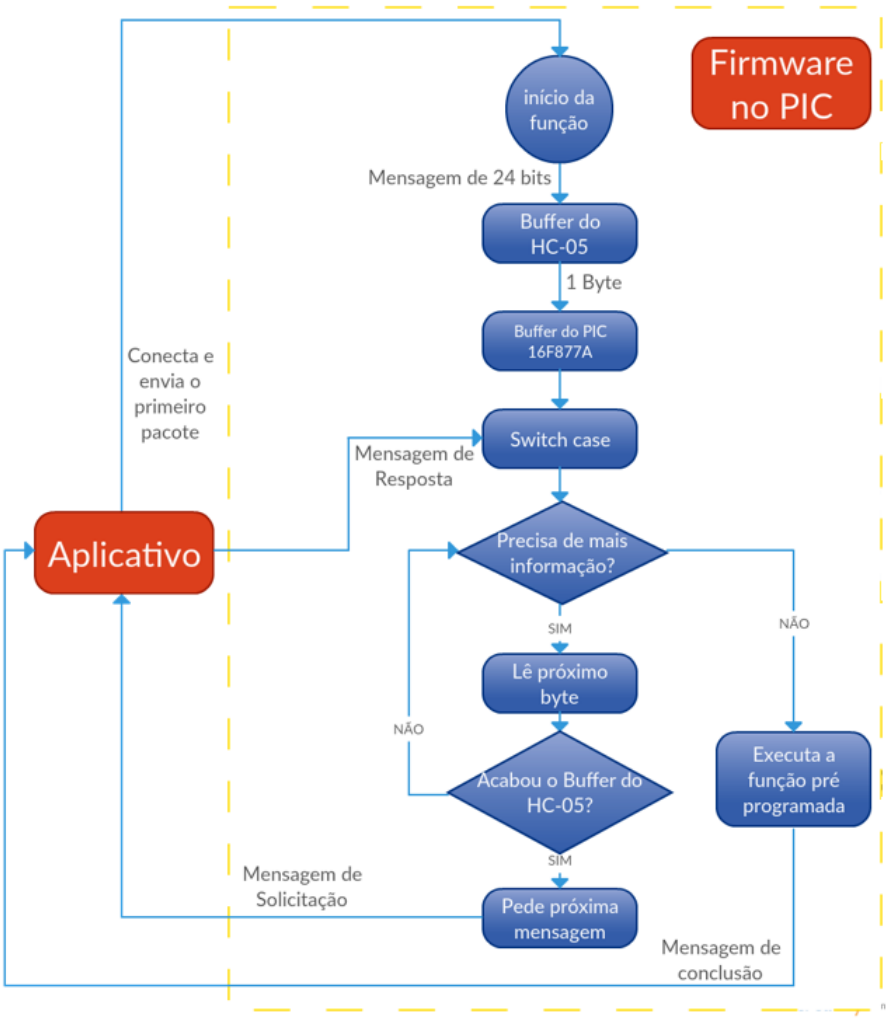

Figura 4: Diagrama do funcionamento do Firmware.

Fonte: *Datasheet PIC16F8877A - Microchip.*

2.4 APLICATIVO *ANDROID*

O aplicativo pertinente ao projeto será desenvolvido para a plataforma *Android* devido a maior acessibilidade desta quando comparada as outras opções, *Windows Phone* ou *IOS*, bem como a familiaridade do grupo com programação na linguagem JAVA, que é a utilizada no ambiente *Android*.

Uma vez escolhida a plataforma *Android*, foi necessário escolher a *IDE* mais adequada ao aplicativo almejado. O grupo estudou as *IDEs* mais populares entre desenvolvedores, particularmente as plataformas *Eclipse* e *Android Studio* e decidiu pela segunda opção, uma

vez que o *Android Studio* foi desenvolvido pela própria *Google*, empresa que desenvolveu o sistema operacional *Android*. Isso proporciona uma maior compatibilidade do compilador com atualizações ao sistema operacional, além de uma estrutura de projetos própria chamada *Gradle* que facilita a criação de múltiplas versões de um mesmo aplicativo.

Com o ambiente de programação escolhido foi preciso conceber uma interface com o usuário intuitiva capaz de receber os comandos do usuário que serão, a seguir, tratados e convertidos para estruturas de dados próprias a transmissão *Bluetooth*. A interface gráfica objetivada e as suas partes estão descritas na Seção 2.4.1, com as classes e estruturas de dados envolvidas no processo detalhadas na Seção 2.4.2.

2.4.1 Interface

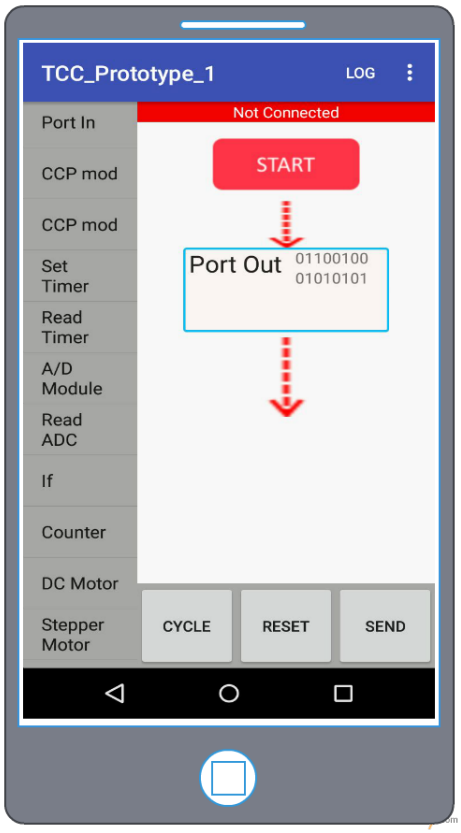

Figura 5: Imagem ilustrativa da interface do aplicativo.

Fonte: Autoria própria.

O *Android Studio* fornece um módulo de desenvolvimento de interface gráfica nativo, que segue uma estrutura em *XML (eXtensible Markup Language)*, fornecendo uma forma textual de definir os elementos da interface gráfica. Essa estrutura tem uma topologia em árvore,

com cada elemento gráfico ocupando uma posição relacionada hierarquicamente aos outros. Os elementos gráficos no ambiente Android são chamados de *views* e podem ser manipulados diretamente através do endereçamento desses *views* em objetos no código. A interface gráfica ilustrada na Figura 5 foi criada a partir de uma série destes *XML*s interpretados pelo *Android Studio*.

A Figura 5 mostra a tela inicial do aplicativo após a inserção de um único bloco de função. Nesta seção apenas são definidas quais as partes essenciais dessa interface e sua função, sendo os diferentes elementos da interface gráfica explorados em mais detalhes no Capítulo 3 deste trabalho.

Os elementos principais da interface gráfica são:

1. Fluxograma de Operação

   Fluxograma criado pelo usuário, representando a sequência de operações a ser seguida pela plataforma microcontrolada. Prevê as funcionalidades de um fluxograma normal, com processos sequenciais simples, comparações lógicas que levam a processos diferentes e operações iterativas (laços de código). Um clique simples em um bloco específico abre uma caixa de diálogo que permite o ajuste dos parâmetros do bloco de função selecionado.

2. Lista de funções

   Lista das funções que o usuário tem ao seu dispor para inserção no fluxograma. Um clique simples adiciona a função selecionada ao final do fluxograma de operação, ao passo que um clique longo abre uma caixa de diálogo contendo uma descrição do funcionamento da função.

3. *Log*

   Qualquer mensagem enviada pela plataforma microcontrolada para o aplicativo será escrito no final do *Log* (não ilustrado na Figura 5). O *log* é simplesmente elemento de texto atualizado a cada nova mensagem recebida pelo dispositivo *Android*

4. Botões de Controle

   Uma série de botões encarregados de funções variadas, como exibir o *log* na tela, estabelecer a conexão com a plataforma microcontrolada, entre outras. Descritos em detalhes no Capítulo 3.

2.4.2 Classes e Estruturas de Dados

Tendo em vista que JAVA é uma linguagem de programação orientada a objeto, uma aplicação desenvolvida na linguagem é formada de classes, a partir das quais se criam estruturas de dados chamadas de objetos, que herdam as características de sua respectiva classe (GOSLING, 2000). No escopo do aplicativo há várias classes com funções bastante distintas, o que nos permite separá-las em grupos, estes sendo:

- Funções

  As funções disponíveis para que o usuário monte seu fluxograma são objetos instanciados a partir de uma mesma classe e armazenado em uma lista de funções. Cada um destes objetos representa uma função específica, com parâmetros ajustáveis pelo usuário.

  A Figura 6 apresenta uma diagrama *UML* simplificado que ilustra a relação entre a lista de funções a cada função individual, sendo os objetos correspondentes as funções objetos do tipo *Function*

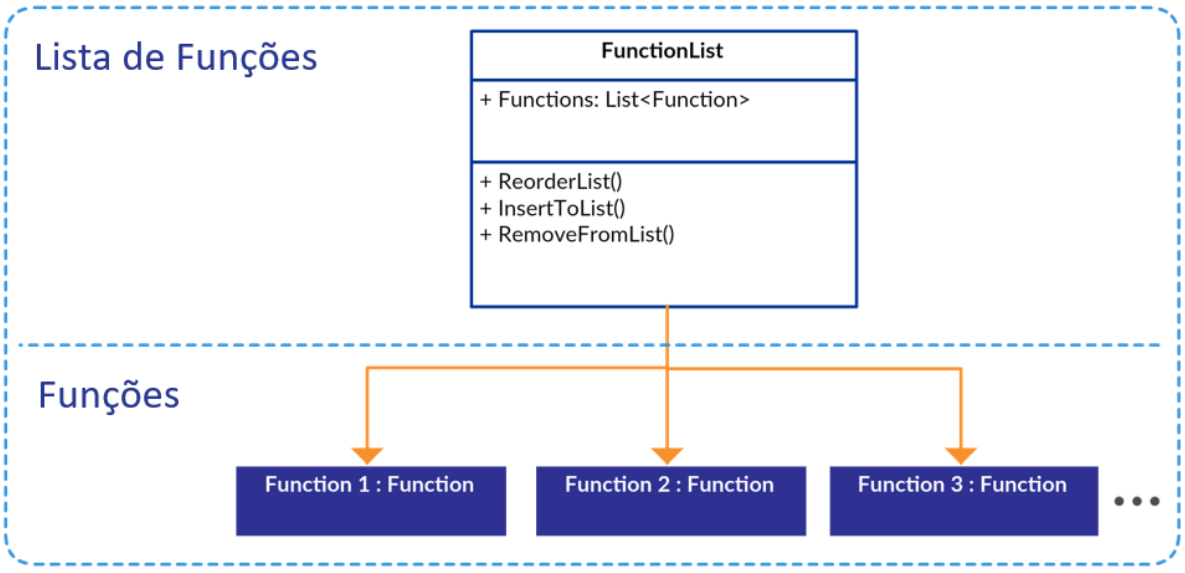

Figura 6: Visualização da estrutura da lista de funções

Fonte: Autoria própria.

- Fluxograma

  O fluxograma em si não é uma classe, mas sim um elemento de interface em *XML* que pode ser referenciado pelo código. Toda vez que o usuário seleciona uma função da lista o aplicativo adiciona um novo elemento no final do fluxograma, tomando a forma de um novo bloco de função.

- Grafo

  O fluxograma da interface gráfica possui um equivalente no código que toma o formato de um grafo, onde cada nó constitui um bloco do fluxograma e cada aresta constitui uma transição entre blocos.

- Tratamento de Dados

  As classes de tratamento de dados tem como função principal traduzir os blocos do fluxograma para um formato adequado para a transmissão *Bluetooth*, sendo que no caso do *PIC16F877A* este formato deve ser um *array* com até 3 *bytes* (limite do *buffer* do módulo HC-05).

  Conforme discutido anteriormente as funções são objetos presentes em uma lista de funções (Figura 7), e cada uma delas possui uma série de variáveis importantes para o funcionamento do aplicativo e para a interpretação de cada função pela plataforma microcontrolada. Dentre as variáveis de cada função a mais importante é o seu ID, que vai ser o primeiro *byte* transmitido para a plataforma microcontrolada, de modo a permitir a identificação da função que esta sendo enviada. Cada objeto do tipo *Function* também possui uma lista de objetos do tipo *Value*, de cumprimento variável de acordo com a função. Essa lista de *Values* é manipulada pelo aplicativo a medida que o usuário constrói e altera o fluxograma. Cada *Value* possui também um ID, identificando a que tipo de valor ele corresponde (valor inteiro, valor binário, valor *byte*, etc.), permitindo que a interface do aplicativo reconheça o tipo de valor e forneça ao usuário metodologias especializadas de modifica-los.

  Durante a transmissão cada objeto do tipo *Value* presente na lista de *Values* de um objeto do tipo *Function* será traduzido para um valor em byte, de modo que uma única função possa formar um *array* de *bytes* a ser enviado para a plataforma microcontrolada, onde o primeiro *byte* é o ID da função e cada byte subsequente é algum valor associado a ela. Caso o *array* de *bytes* a ser enviado seja de um tamanho maior do que o *buffer* da antena utilizada pelo sistema (24 bits para o caso do módulo HC-05 utilizado neste projeto), o aplicativo divide o *array* em multiplos sub-*arrays*, enviando um por vez e só enviando o próximo quando a plataforma microcontrolada confirmar o recebimento. Esse processo é ilustrado na Figura 7, onde apresentamos um diagrama de relação de objetos simplificado, de modo a ilustrar a construção dos *arrays* de *bytes* enviados.

- Comunicação *Bluetooth*

  O pareamento de dispositivos é realizado na *Thread* principal do aplicativo. Entretanto, todas as outras operações relacionadas a transmissão de dados precisam ser realizadas por

uma segunda *Thread*, de modo a não bloquear a interface gráfica do aplicativo quando este está aguardando mensagens da plataforma microcontrolada. Essa *Thread* de comunicação toma a forma de uma classe. Essa classe tem uma referência ao grafo correspondente ao fluxograma e percorre esse grafo nó por nó. A *Thread* obtém as funções correspondentes a cada bloco, realiza o tratamento dos dados de cada uma e finalmente realiza o envio do *array* de *bytes* correspondente. Toda vez que a *Thread* termina o envio de um bloco de função ela entra em modo de escuta, aguardando mensagens da plataforma microcontrolada que peçam por um novo envio.

- Controles

  O controle é um objeto compartilhado pelas *Threads* do aplicativo. Ele contem uma série de variáveis booleanas e inteiras, que servem para a comunicação entre as *Threads*.

2.5 COMUNICAÇÃO

Para realizar a comunicação entre o aplicativo *Android* e a plataforma microcontrolada utilizou-se o protocolo de comunicação *Bluetooth*. O maior motivador dessa escolha é a presença quase universal de ferramentas capacitadas para este tipo de comunicação em dispositivos *Android*. Além disso, os membros da equipe já utilizaram o protocolo previamente, estando razoavelmente familiarizados com sua implementação no ambiente *Android*.

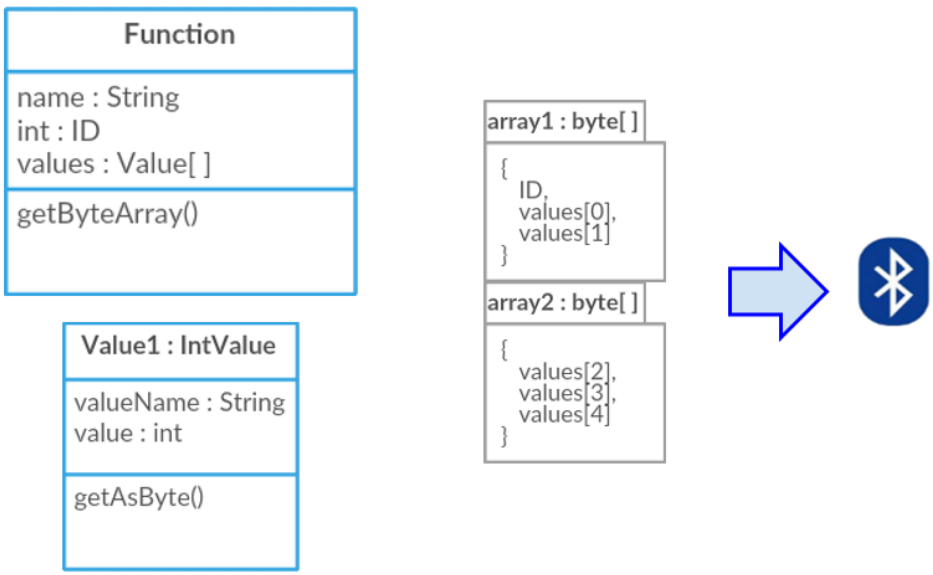

Figura 7: Estruturação dos pacotes.

Fonte: Autoria própria.

Tendo em vista a escolha da comunicação *Bluetooth*, nesta seção há detalhes de como ela será implementada no aplicativo e como se da o recebimento dos dados pelo *Bluetooth* acoplado a plataforma microcontrolada.

2.5.1 Implementação no Ambiente *Android*

O ambiente *Android* fornece uma série de bibliotecas para implementar a comunicação *Bluetooth*, sendo possível resumir o processo em três passos gerenciados pelo dispositivo *Android*:

1. Pareamento de Dispositivos

   Nessa etapa o aparelho *Android* procura dispositivos *Bluetooth* que estejam operando no modo *discoverable*, ou seja, visíveis para outro aparelhos e estabelece o pareamento entre os dispositivos. Essa etapa pode ser ignorada caso os dispositivos já tenham sido pareados previamente.

2. Obter Dispositivos Pareados

   Esta etapa consiste simplesmente na obtenção da lista de dispositivos pareados com o aparelho *Android* e no seu mapeamento em uma variável do tipo *Lista* dentro do aplicativo.

3. Estabelecer Conexão

   Nesta etapa se cria um canal de radio-frequência adequado a comunicação entre o aparelho *Android* e o módulo *HC-05*, o que permite a transmissão de dados.

No cenário do projeto primeiro será realizada uma busca por dispositivos ativos próximos ao aparelho *Android*. Se o aparelho encontra um dispositivo ativo ele inicia o processo de conexão, abrindo um canal de comunicação rádio-frequência e atribuindo a este uma chave de identificação única. Estabelecido este canal de comunicação o envio de dados pode ser iniciado dado o comando do usuário. Este processo é gerenciado pelo aplicativo *Android* desenvolvido neste trabalho, e o modulador *HC-05* está descrito na Seção 2.5.2 deste trabalho.

2.5.2 Módulo *Bluetooth*

O *Kit Micropic* não possui circuito de suporte à comunicação via *Bluetooth*. Para que seja possível realizar a comunicação entre o dispositivo móvel e a placa de circuitos utilizaremos

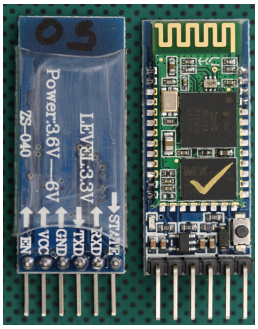

Figura 8: Modulador Bluetooth HC-05 ZS-040.
Fonte: HC Serial Bluetooth Products User Instructional Manual.

a *UART* do *PIC16F877A*, optamos pelo uso de um módulo externo capaz de converter o sinal de rádio para comunicação serial a nível *TTL* que possa ser lida pela *UART* do microcontrolador. O *HC-05* foi o módulo de preço acessível encontrado pela equipe, mas as suas especificações e funcionalidades, como por exemplo comandos internos que permitem definir a taxa de *baud* e o nome do dispositivo a ser reconhecido na conexão *Bluetooth*, são as necessárias para que o mesmo possa ser utilizado no projeto.

2.6 FLUXO DA APLICAÇÃO

Para possibilitar uma visão mais detalhada do projeto é mostrada uma representação do fluxo da aplicação em formato de fluxograma, que pode ser visto na Figura 9. O processo pode ser resumido em quatro etapas: duas no escopo do aplicativo *Android* e duas no escopo da plataforma microcontrolada. A execução destas quatro etapas constitui um ciclo que é o processo de envio, recebimento, execução e confirmação da execução da função de um único bloco do fluxograma.

As etapas do funcionamento do aplicativo são:

1. Geração de Dados

   O aplicativo começa nesta etapa, com o usuário construindo o fluxograma de funções da aplicação. Uma vez construído o fluxograma serve como referência para a obtenção dos blocos de função a serem enviados para a plataforma microcontrolada. Esta etapa termina com a determinação da próxima função a ser enviada, ilustrada na Figura 9 no bloco 'Função Atual'

2. Tratamento Pré-Transmissão

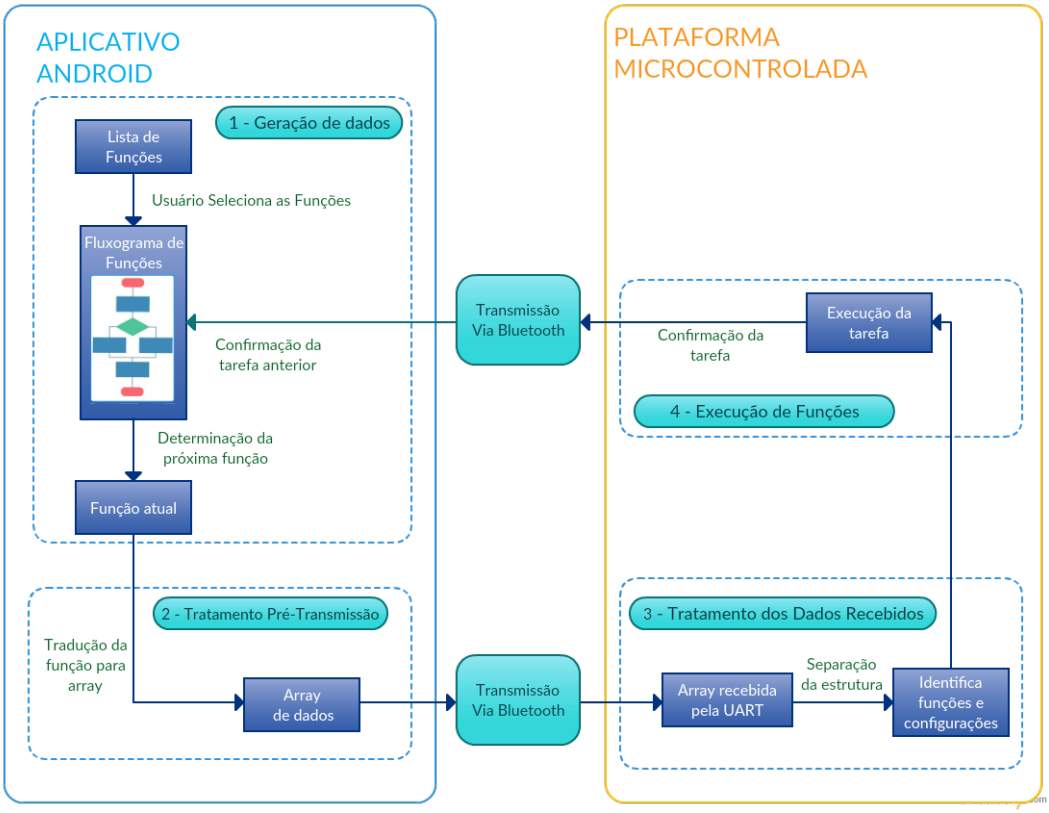

Figura 9: Fluxograma de Operação do Projeto.

Fonte: Autoria própria.

Nesta etapa o aplicativo deve traduzir as informações do bloco de função a ser enviado, incluindo os valores dos parâmetros definidos pelo usuário, para um formato adequado a transmissão. No caso do *PIC16F877A* este formato é um *array* contendo até três *bytes*, que é enviado para o módulo *HC-05* da plataforma microcontrolada.

3. Tratamento dos Dados Recebidos

O módulo HC-05, uma vez tendo recebido os dados enviados do aparelho *Android*, se encarrega de transformar o sinal em níveis *TTL* no formato *ASCII* que possam ser interpretados pela *UART* do microcontrolador por conexão de suas portas seriais.

Tendo obtido os dados (recebidos no formato de um *array* de até 24 *bits*) ainda é necessário processar as informações nele presentes. Esse processo vai separar as informações pertinentes a cada etapa de configuração da atual função no microcontrolador. Cada pacote enviado pode conter até 24 *bits* sendo armazenado no *buffer* do módulo *HC-05* e lido de *byte* a *byte* pela *UART*. O primeiro *byte* refere-se sempre à função a ser executada, funcionando como um ID e os dados subsequentes variam em significado de acordo com o que for necessário para a configuração desta

mesma função.

4. Execução de Funções

   Com as informações obtidas do código do *array* inicia-se a execução da função atual do fluxograma. Assim que a tarefa for concluída, será enviada uma confirmação por meio do sinal *Bluetooth* para o o aplicativo *Android*. O ciclo continuará retornando ao item 1, obtendo o próximo bloco de função a ser enviado a partir do fluxograma criado pelo usuário, até que todo o fluxograma seja percorrido, ou que o processo seja interrompido pelo usuário.

3 OPERAÇÃO

Para o funcionamento correto do sistema, tanto no escopo do aplicativo quanto no da plataforma microcontrolada, alguns cuidados operacionais são precisos. Neste capítulo serão descritos alguns dos procedimentos necessários no pré projeto, seguido de uma descrição mais detalhada da operação do aplicativo desenvolvido.

3.1 PRÉ PROJETO

Para o funcionamento completo da plataforma vê-se necessário: ter o aplicativo instalado em um dispositivo *Android* equipado com *Bluetooth*, a execução do mesmo durante todo o tempo de uso da plataforma, a alimentação e montagem do circuito microcontrolado de acordo com o uso desejado e o estabelecimento da comunicação entre as duas partes do sistema. Na sequência deste capítulo serão discutidas cada uma destas condições.

3.1.1 Instalação do Aplicativo

O aplicativo será distribuído na forma de um arquivo *APK*. O sistema operacional de dispositivos Android reconhece esse tipo de arquivo e abre um *dialog* quando este é recebido, permitindo que o usuário confirme a instalação. O sistema *Android* se encarrega do resto do processo.

3.1.2 Alimentação do Sistema

No caso do *PIC16F877A* utilizado nas aplicações desenvolvidas no projeto a alimentação foi uma fonte CA/CC com saída entre 9V/1A e + 12V/1A com conector do tipo P4 (2,5 x 5,5 mm) ou baterias com especificações de saída semelhantes.

3.1.3 Estabelecendo a comunicação

Antes de entrar no escopo do aplicativo é necessário realizar o pareamento do aparelho *Android* com o módulo HC-05. Uma vez alimentado o *HC-05* pode ser encontrado pelas

funcionalidades de pareamento encontradas na grande maioria dos aparelhos *Android* com compatibilidade *Bluetooth*.

Feito o pareamento cabe ao aplicativo estabelecer a conexão que permita a transmissão de dados. Isso é realizado através do botão *'Connect'* do sub-menu da barra superior do aplicativo (descrito na Seção 3.2) seguido da seleção do HC-05 no *dialog* aberto. Feita a conexão o aplicativo está preparado para começar a comunicação com a plataforma microcontrolada.

3.2 OPERAÇÃO DO APLICATIVO

A plataforma microcontrolada tem como principal objetivo tornar simples e acessível a programação e execução de projetos para qualquer usuário. Conforme discutido a forma de interação com o usuário escolhida para isso foi a montagem de um fluxograma. Para a manipulação deste fluxograma, além do controle dos processos de comunicação do aplicativo, foi criada a interface gráfica apresentada na Figura 10.

O fluxograma de operação é criado por meio da adição de funções por meio de sua seleção da lista de funções (item 1 da Figura 10), e a configuração dos parâmetros da função é realizada por meio de um clique no bloco de função criado na região do fluxograma (item 3 da Figura 10). Quando o fluxograma de operação estiver pronto, e o dispositivo estiver conectado ao módulo *HC-05*, basta iniciar o envio das instruções pressionando o botão '*Send*' (item 4 da Figura 10, botão inferior direito).

Segue uma descrição mais detalhada dos itens da Figura 10:

1. Lista de Funções

   Conforme descrito em seções anteriores a lista de funções contém todas as funções disponíveis para a montagem do fluxograma. Um clique simples em qualquer uma das funções cria um bloco de função no espaço do item 2, e um clique longo abre uma caixa de diálogo contendo uma breve descrição da função.

2. Espaço para Bloco de Função Seguinte

   Após um clique simples em uma função da lista o bloco de função é criado no próximo espaço vazio do fluxograma. Quando começa a comunicação com a plataforma microcontrolada edições do fluxograma são proibidas, e um bloco marcando o final do fluxograma é criado neste local.

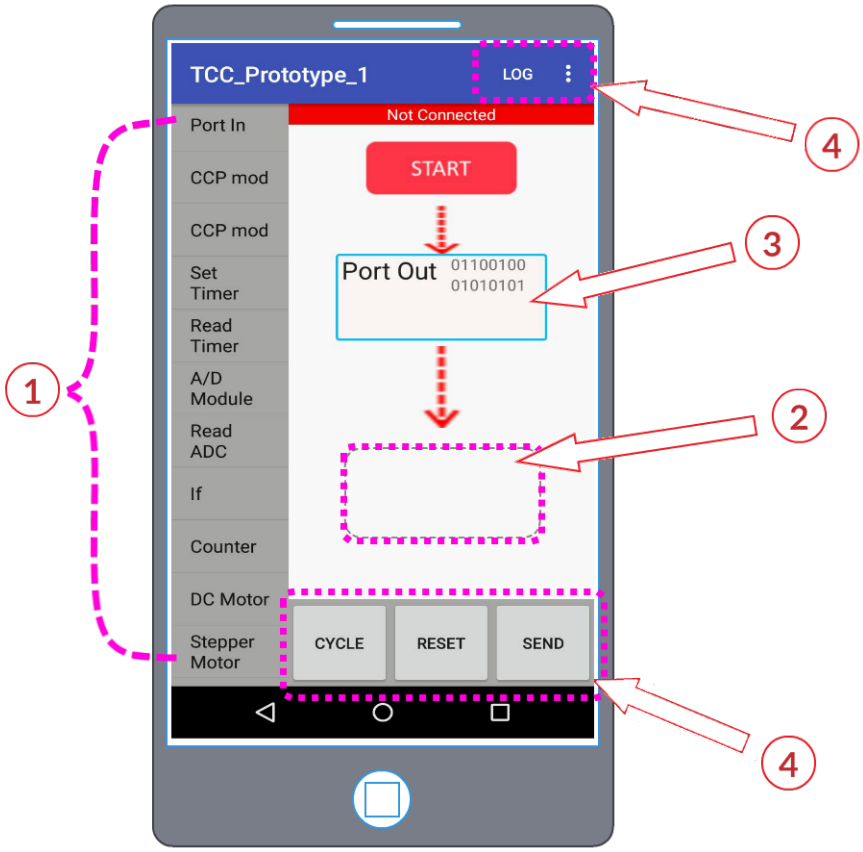

Figura 10: Imagem ilustrativa da interface do aplicativo com marcações.

Fonte: Autoria própria.

3. Bloco de Função

   Representação gráfica de um bloco de função do fluxograma, incluindo o nome da função no na esquerda do bloco e os valores dos seus parâmetros na direita do bloco. No exemplo da figura a função é a *Port Out*, descrita no Capítulo 4, cujo parâmetros são *Port* e *Value* sendo o primeiro a porta de destino (A, B, C, D ou E para o *PIC16F877A*), e o segundo o valor que a porta deve assumir. Um clique simples abre um *dialog* que mostra o nome de cada parâmetro e permite a alteração dos valores associados, com uma interface customizável de acordo com a função.

4. Botões

   Os botões ao programa tem várias funções diferentes, e estão separados em dois grupos: os da barra de ação (parte superior do aplicativo) e os da barra de navegação (parte inferior do aplicativo).

   Barra de Navegação:

(a) Botão *Cycle*

Este botão está encarregado de controlar as características cíclicas do fluxograma. Após clicar no botão o usuário deve selecionar um bloco de origem, seguido da seleção de um bloco de destino. O fluxograma é alterado de acordo com as seleções do usuário.

(b) Botão *Reset*

O botão de *reset* limpa o fluxograma da tela e todos os objetos relacionados a ele no código. Além disso ele também zera a conexão com o HC-05.

(c) Botão *Send*

Cria a *Thread* de comunicação e começa as rotinas de comunicação com a plataforma microcontrolada.

Barra de Ação:

(a) *Log Toggle*

Este botão funciona como um comutador. Quando clicado pela primeira vez ele troca a visibilidade da área do fluxograma, substituindo-a por um *log* textual que exibe todas as mensagens recebidas da plataforma microcontrolada. Um segundo clique esconde o *log* e exibe o fluxograma novamente.

(b) Submenu

O submenu contem dois itens:

O primeiro é o *Connect*, que quando selecionado abre um *dialog* que permite que o usuário selecione um dispositivo ao qual se conectar.

O segundo é o *Settings*, que abre uma tela apresentando algumas variáveis de controle do aplicativo, contemplando, por exemplo, a seleção do tipo de microcontrolador utilizado na plataforma microcontrolada, ou a antena utilizada. O aplicativo no seu estado atual tem suporte apenas para o *PIC16F877A* e para o modulador *HC-05*.

4 FUNÇÕES DISPONÍVEIS

A biblioteca de funções apresentada ao usuário na aba do lado esquerdo do aplicativo é a forma fundamental da interface entre o usuário e o sistema. Cada uma delas representa operações variadas, das mais básicas até as de maior nível possibilitando transições entre programação de maior controle e maior agilidade. Cada função é representada como um retângulo com seu mnemônico e deve ser transferida para a área do gráfico para então ser parte do programa e, a partir disso, ser configurada de acordo com a aplicação desejada.

As funções são divididas entre funções básicas e *macro* funções. Sendo o segundo grupo aplicações que envolvem múltiplas funções do primeiro grupo para o cumprimento de tarefas mais complexas. A seguir são listadas as funções e detalhes de sua aplicação e configuração bem como um resumo do que fazem.

4.1 FUNÇÕES BÁSICAS

Estas funções são as mais fundamentais para o programa e operação do microcontrolador, são em geral funções que manipulam diretamente os valores dos registradores do microcontrolador e também afetam diretamente os periféricos do próprio, não sendo necessários circuitos específicos para aplicar as saídas desejadas.

- TRIS

  A função *TRIS* configura os registradores *TRISA*, *TRISB*, *TRISC*, *TRISD* e *TRISE* do *PIC* que são responsáveis pelo comportamento dos pinos *I/O*. Ao adicionar a função na área do fluxograma, o quadro correspondente é criado no fluxograma.

  Para acessar as configurações, é necessário selecionar o quadro TRIS. Ao clique surgirão as opções para selecionar o registro a ser alterado e uma sequencia de *checkboxes* representando os pinos controlados por este registro. Os pinos selecionados com a *checkbox* preenchida serão configuradas como entradas e os de *checkbox* vazia serão configurados como saídas do microcontrolador.

  O registrador *TRISE* é um caso especial, os *bits* de configuração de 4 a 7 configuram funções da porta paralela ainda não implementada no escopo deste projeto. Portanto,

altere apenas os bits 0 a 2 para controlar esta função.

- PORT OUT

  Muito semelhante à anterior ao respeito da configuração, a função *PORT OUT* é usada para alterar o estado lógico das portas de saída controladas por meio da alteração dos registros *PORTA*, *PORTB*, *PORTC*, *PORTD* e *PORTE*. Ao adicionar a função na área do fluxograma, o quadro correspondente é criado.

  Para acessar as configurações, selecione o quadro PORT OUT. Ao clique surgirão as opções para selecionar o registro a ser alterado e uma sequencia de *checkboxes* representando os pinos controlados por este registro. Os pinos selecionados com a *checkbox* preenchida serão configuradas como saídas ativas e os de *checkbox* vazia serão configurados como saídas inativas.

  É importante ressaltar que apenas os pinos configurados como saídas do sistema podem ser manipulados por esta função. Para configurar as entradas e saídas do sistema, execute primeiramente a função TRIS no programa para garantir o funcionamento desejado.

- PORT IN

  Função utilizada para obter o estado lógico do pino ou da porta desejada. Lendo o *bit* ou *byte* selecionado, é possível tomar decisões dentro de seu código ou guardar dados para uso posterior.

  Para acessar as configurações, selecione o quadro *PORT IN* no espaço de programa. Ao clique surgirão as opções para selecionar o registro a ser lido e uma segunda opção para selecionar se a porta inteira deve ser lida ou apenas um dos pinos.

  Apenas os pinos configurados como entradas do sistema podem ser lidos com esta função. Para configurar as entradas e saídas do sistema, execute primeiramente a função *TRIS* no programa para garantir o funcionamento desejado.

- Esperar

  Sempre que necessário, o circuito pode manter seu estado atual e apenas deixar que se passe um período limitado de tempo. Para este objetivo a função Esperar pode ser usada para manter o sistema neste estado pelo tempo selecionado em segundos

  Para acessar as configurações, selecione o quadro Esperar no espaço de programa. Ao clique surgirá um campo para números, o número escrito pelo usuário será a duração da pausa do sistema em segundos.

- TIMER

  Essa função se destina a configurar os diferentes *Timers* do *PIC*, cada *Timer* tem seu próprio registro de configuração, dentro do quadro de configuração desta função é possível escolher o *Timer* a ser configurado e os detalhes de funcionamento específicos.

- Leitura do timer

  Utilizada para retornar o atual valor do *timer* selecionado para o aplicativo para ser utilizado como variável ou apenas mostrada em *LOG*.

  No quadro de configuração, apenas selecione qual o *timer* a ser lido pelo sistema.

- ADC

  O módulo *ADC* do *PIC* é configurado a partir de dois registradores, *ADCON0* e *ADCON1*, cada registrador deverá ser configurado separadamente. O registrador *ADCON1* configura o divisor do *Clock* a ser usado para as conversões

  O registro *ADCON* configura os pinos entrada analógica e entrado ou saída digital do conversor, referentes a porta A do *PIC*. A configuração deste registro ainda depende da configuração do registro *TRISA* para definir as entradas e saídas do sistema.

- Leitura ADC

  Função destinada a resgatar o valor atual da porta ligada ao módulo *ADC* do *PIC*. Para ler o valor do canal desejado, selecione-o no menu de configuração da área de programa a informação requisitada é mostrada em *LOG*.

- Escolha

  Esta função cria uma bifurcação no programa, é possível escolher como o sistema escolherá o próximo passo por meio do menu de configuração da função.

  As opções existentes de comparação são maior, menor, igual ou diferente e as variáveis a serem comparadas são os pinos de saída de qualquer porta do *PIC*, os valores de variáveis internas do aplicativo, ou ainda valores fixos definidos pelo usuário.

- Contador

  São disponibilizados dois contadores digitais para que possam ser utilizados nas mais diversas aplicações pelos usuários. Esta função pode incrementar ou decrementar estes contadores de acordo com desejo do usuário.

  Para realizar estas manipulações selecione o contador e a opção desejada no menu de configuração.

- Reset/Master Clear

  Esta função reinicia a *CPU* do microprocessador, provisoriamente programado em *firmware* para ativar o relé eletromecânico do *Micropic* fechando o contato entre o pino *MCLR* do *PIC* e o terra.

- Wait For It

  Função utilizada para esperar por uma condição específica para continuar a execução do programa. Os argumentos a serem introduzidos são os dois valores a serem comparados e a condição a ser testada. A função entrará em *loop* até que a condição seja verdadeira, após isso o programa continuará sua execução de acordo com o fluxograma.

4.2 MACRO FUNÇÕES

São funções que abrangem um maior numero de alterações nos registradores, seja em quantidade ou sequência de manipulações. Tornam o controle das funções mais rápido e fácil, mas alienam o usuário de parte do processo. Estas são principalmente voltadas para operar os circuitos e aplicações externas ao microcontrolador, portanto algumas necessitam de circuitos e conexões específicas que serão detalhados a seguir.

- Motor DC

  Destinada ao acionamento de motores de corrente contínua por meio da ponte-H instalada no *MICROPIC*. Quando um ou dois motores são ligados à placa pelo conector P13, é possível acioná-los instantaneamente por meio desta função.

  Quando selecionada, esta função permite escolher os motores a serem acionados e o sentido do giro.

- Motor de passo

  Criada para operar motores de passo monopolares conectados ao conector P8 do *Micropic*. As configurações permitem escolher o sentido de giro e o número de passos desejados.

- PWM

  O *PIC16F877A* possui dois módulos *CCP* dos quais apenas o modo *PWM* será controlado no escopo deste trabalho. Uma vez que a configuração deste periférico precisa de alterações em diferentes registros.

Alterações neste quadro podem alterar configurações anteriores dos registros *TRISC* referente a configuração de entradas e saídas da porta C do *PIC*, *T2CON* referente ao *timer 2* e *CCP1CON* e *CCPR1L* referentes ao próprio modulo *CCP*.

A saída do sinal *PWM* ocorrerá pelo pino C2 caso seja configurado o módulo *CCP1* e pelo pino C1 caso seja configurado o modulo *CCP2*.

- Cont. Car. Esta é uma função específica para a configuração da aplicação exemplo do carro da Seção 5.1 deste documento. Os argumentos de entrada nesta função se referem à movimentação desejada e são: 'W' para ligar o motor para avançar com o carrinho, 'S' para para-lo, 'A' para virar à esquerda e 'D' para virar à direita.

5 APLICAÇÕES EXEMPLO

Com objetivo de demonstrar a facilidade de criação no sistema desenvolvido, bem como a flexibilidade de seu uso, foram criados dois projetos exemplo a serem descritos na sequencia do capítulo.

5.1 MODELO DE CARRO DE TRAJETÓRIA PRÉ-DEFINIDA

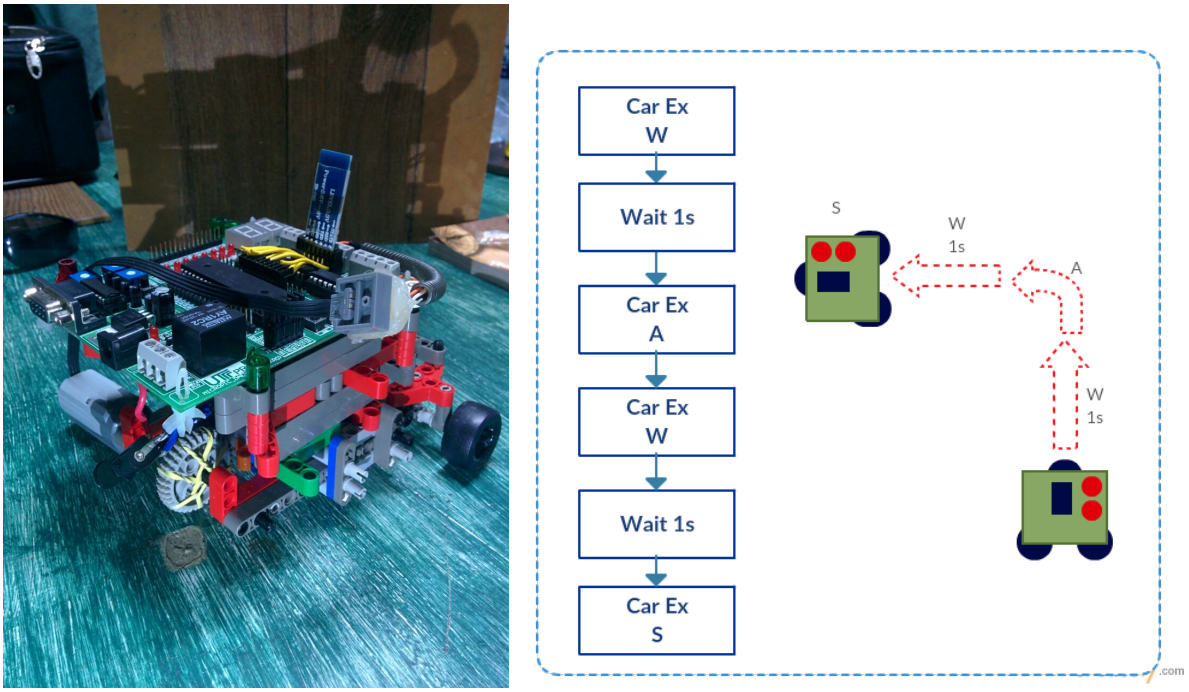

Figura 11: Modelo de carro e exemplo de programa.

Fonte: Autoria própria.

Por meio da operação de um motor de corrente contínua e um motor de passo bipolar, foi criada a estrutura de um carro de três rodas com peças de LEGO, mostrado a figura 11. Este modelo busca demonstrar de forma visual o sequenciamento de ações pelo usuário e aproxima da maneira mais básica o conceito de programação de uma trajetória. O circuito de comando aplicado está demonstrado na Figura 12.

Por meio das funções “Car Ex.”e *“Wait”*, o usuário é capaz de programar uma trajetória

definida para o modelo de carrinho acompanhando a execução do programa pelo aplicativo e pelo próprio movimento do sistema microcontrolado pela superfície. Simulando a configuração direcional em teclados, os argumentos passados nesta função são “w” para avançar o carro, “s” para parar o motor, “a” para virar à direita e “d” para a esquerda.

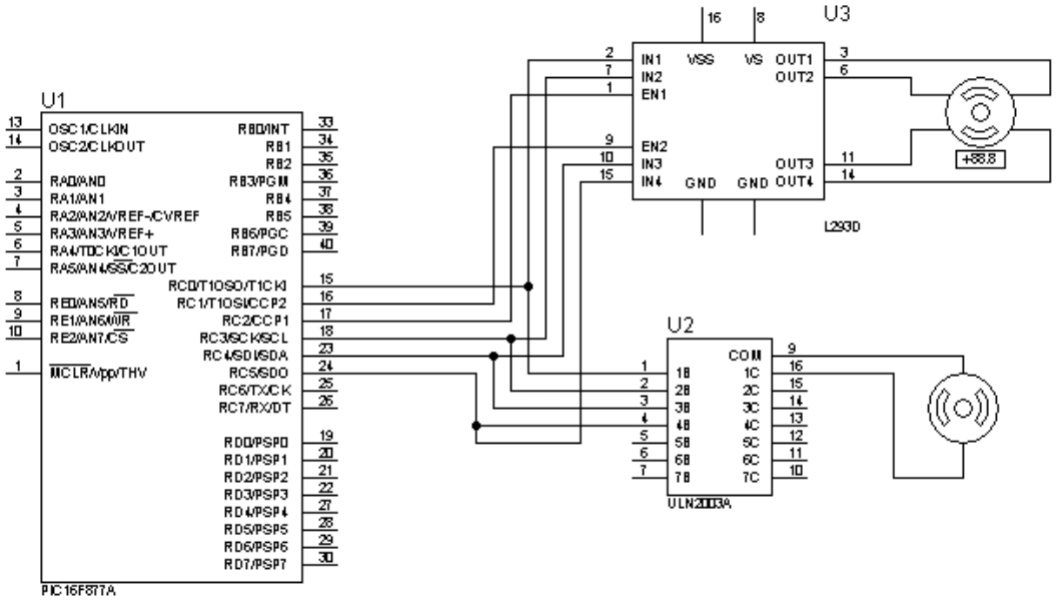

Figura 12: Diagrama simplificado do circuito de controle do carro.

Fonte: Autoria própria.

A Função “Car Ex.”é um exemplo de macro função a ser criada pelo usuário como um programa completo salvo como novo bloco de função, uma representação para futuras aplicações, discutidas na Seção 6.1.

O resultado da aplicação com o circuito sugerido e a estrutura mostrada na Figura 11 foi que o carrinho respondeu em tempo esperado aos comandos do aplicativo. Devido ao fato do piso em que o teste ocorreu ser muito liso, as curvas não foram precisas. O fato de que o carrinho estava sendo alimentado por um fio também restringiu seu movimento, no entanto houve a comprovação do conceito, o melhor funcionamento desta aplicação é dependente de melhor *design* do modelo, não sendo este parte do escopo deste trabalho.

5.2 ALARME DE PRESENÇA

Para demonstrar como o sistema pode operar gerenciando entradas e saídas do sistema, foi criado um modelo simplificado de alarme com luz infra-vermelha. Neste projeto-exemplo,

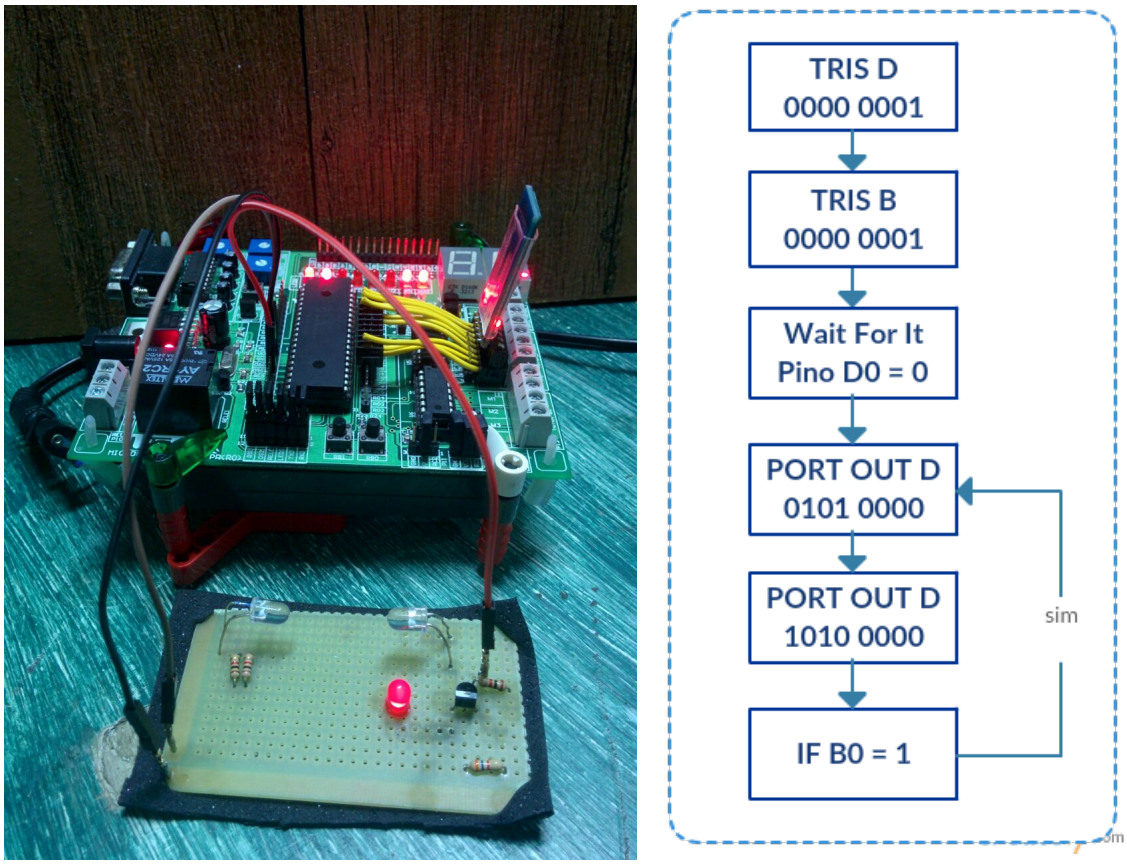

Figura 13: Exemplo de alarme de presença.

Fonte: Autoria própria.

quando o sinal do *LED* infra-vermelho é interrompido, um alarme dispara, simbolizado por um dos *LED*s do *Micropic* e pode ser rearmado quando o botão *S3* do *kit* é pressionado. O circuito está representado na Figura 14.

As funções utilizadas são TRIS, PORT OUT, WAIT FOR IT, IF e WAIT. A determinação das entradas e saídas do sistema de acordo com a Figura 14 estão a cargo da preferência do usuário, pois a figura apenas demonstra a lógica da montagem necessária.

Com a implementação do circuito sugerido, foi possível armar e desarmar o alarme uma vez, comprovando que o sistema é capaz de criar uma aplicação com comportamento reativo. Assim que o programa foi enviado, o sistema já estava pronto para reagir ao interrompimento do acoplamento ótico. Assim que o sinal foi interrompido, o alarme disparou rapidamente. O botão para desarmar o alarme precisou ser acionado no exato momento da leitura no *loop* do alarme, deixando espaço para melhoramentos futuros no programa, como uma função de interrupção para o microcontrolador.

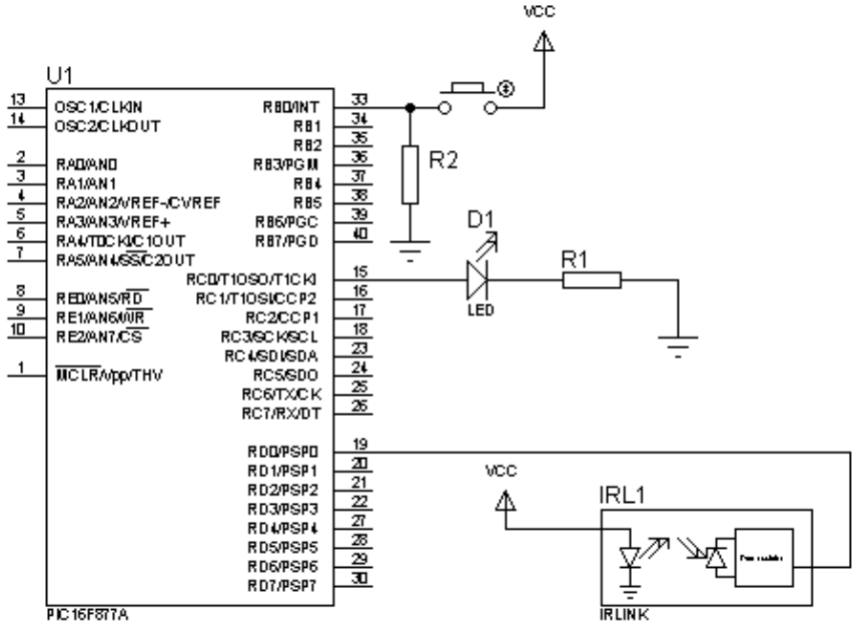

Figura 14: Diagrama simplificado do circuito de controle do alarme.

Fonte: Autoria própria.

6 CONCLUSÃO

Ao longo deste trabalho foram descritas as várias partes que compõem a plataforma de programação desenvolvida, sendo esta composta pelo aplicativo *Android*, a plataforma microcontrolada e os mecanismos de comunicação. Além disso foram apresentadas duas aplicações exemplo e as funções disponíveis implementadas pela equipe. Esses componentes trabalhando em conjunto forneceram uma plataforma eficiente para o desenvolvimento de pequenas aplicações de sistemas microcontrolados que possam ser estruturadas no formato de fluxograma, tendo como principal vantagem a facilidade de montar o programa de operação sem a necessidade de alterar o *firmware* do microcontrolador. A comunicação em tempo real do aplicativo com o sistema microcontrolado permitiu que o *PIC* respondesse aos comandos enviados pelo celular (na forma de blocos de função) e executasse a função pertinente imediatamente. Conforme demonstrado pela primeira aplicação exemplo, o sistema foi adequado para controlar a movimentação de um pequeno carro elétrico ligado a um motor DC e um motor de passo, servindo como prova de conceito do projeto.

Os objetivos propostos na Seção 1.1 deste documento foram todos concluídos e o tempo de resposta do sistema é satisfatório para as aplicações sugeridas. As principais limitações do sistema estão no limite de memória *flash* do *PIC*, a falta de capacidade de executar mais de uma *thread* ao mesmo tempo no microcontrolador e o tamanho do *Buffer* do módulo *HC-05*. Outra limitação significativa do aplicativo foi o fato de o fluxograma resultante não ser tão versátil quanto o objetivado no começo do trabalho. Isso foi devido principalmente a dificuldade da equipe em encontrar bibliotecas adequadas para representar o fluxograma na tela do celular. O grafo correspondente ao fluxograma existe com uma implementação desenvolvida pela própria equipe e possui a estrutura necessária para a formação de qualquer fluxograma, entretanto o processo de manipulação da interface gráfica para desenhar o grafo dentro das limitações do aplicativo se provou muito custoso em termos de tempo de trabalho, justificando a possibilidade futura de adaptar o sistema para uma máquina de estados ao invés do grafo utilizado.

A plataforma se encontra em estado de nascença, possuindo apenas algumas poucas funções disponíveis, devido a dificuldade de teste de cada função e das limitações do microcontrolador utilizado (cuja memória FLASH está quase completamente preenchida com

o *firmware* atual, que contem apenas 16 funções). Mesmo a interface gráfica possui grandes margens para melhoras, sendo bastante simples até o momento. Entretanto, a plataforma já se provou de grande utilidade para o desenvolvimento de pequenas aplicações microcontroladas, conforme provado nas duas aplicações exemplo. As possibilidades da plataforma quando se discute as perspectivas futuras (Seção 6.1) são muitas e o sistema desenvolvido parece promissor para utilização nas áreas discutidas na introdução deste trabalho, como em sala de aula ou para uso de hobbyistas. Desta forma, a plataforma promete ser uma ferramenta simples e conveniente para a criação de pequenos projetos eletrônicos, conforme o objetivado na concepção do projeto.

6.1 PERSPECTIVAS FUTURAS

Mesmo concluídos os objetivos propostos para o desenvolvimento do trabalho de conclusão de curso, existem ainda muitos viés de expansão. A ideia da plataforma de desenvolvimento pode ser adaptada a diversos outro microprocessadores por meio de *firmware*s específicos expandindo as possibilidades e o público alvo.

Também existe a possibilidade de criar um serviço de compartilhamento de códigos por meio de códigos *QR* referenciando um banco de dados online. A perspectiva de compartilhamento de macro-funções entre usuários, distribuídas através da internet, expande os horizontes da plataforma. Caso programas criados dentro do aplicativo sejam salvos e utilizados dentro de outros códigos como macro-funções personalizadas, programas mais complexos poderiam ser criados por meio da colaboração de diversos autores.

Há também a possibilidade de criar placas de circuito melhor adaptadas para fácil instalação com os mais diversos conectores como entradas *RJ-45*, conectores para cabo-banana ou inspirados em soquetes de memórias *DDR3* permitindo ainda mais praticidade na implementação de diferentes projetos.

REFERÊNCIAS

BLUETOOTH, S. Bluetooth core specification version 4.0. **Specification of the Bluetooth System**, 2010.

CARELLE, E. **PKBurner**. 2016.

GOODMAN, B. G. **Background code update for embedded systems**. Google Patents, ago. 1 2006. US Patent 7,086,049.

GOSLING, J. **The Java language specification**. Addison-Wesley Professional, 2000.

HI-TECH, C. **Compiler, Microchip Technology Inc, AZ, 2010**. 2010.

INSTRUMENTS, T. Data sheet max232. **Online Document**, v. 15, 2004.

INSTRUMENTS, T. **L293D datasheet,"Quadruple Half-H Drives**. 2010.

JONES, D. W. Basic stepping motor control circuits. **Stepping Motors**, v. 2004, 2007.

KOBAYASHI, C. Y. A tecnologia bluetooth e aplicações. **USP. São Paulo**, 2004.

LIAN, K.-Y.; HSIAO, S.-J.; SUNG, W.-T. Mobile monitoring and embedded control system for factory environment. **Sensors**, v. 13, n. 12, p. 17379, 2013.

MCROBERTS, M. **Arduino básico**. Novatec Editora, 2011.

MICROCHIP, D. P. **Datasheet PIC 16F877**. 2013. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf>.

MPLAB, I. Quick start guide. **Microchip Technology Inc**, 2006.

REUTLINGEN-UNIVERSITY. **HC Serial Bluetooth Products User Instructional Manual**. 2016. Disponível em: <http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05$_BT - Modul.pdf$¿.

SOURCEFORGE. **Tiny Multi Bootloader+**. 2016. Disponível em: <https://sourceforge.net/projects/tinypicbootload/>.