UNIVERSIDADE FEDERAL DO PIAUI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

PROTÓTIPO MODULAR DO SISTEMA DE INTERCOMUNICAÇÃO ARDUINO - RASPBERRY PI (ArPI)

FLÁVIO DOS SANTOS ESCOBAR

ORIENTADOR(A): Prof. Ivenilton Alexandre de Souza Moura

PICOS - PI
2016

FLÁVIO DOS SANTOS ESCOBAR

PROTÓTIPO MODULAR DO SISTEMA DE INTERCOMUNICAÇÃO ARDUINO - RASPBERRY PI (ArPI)

Monografia submetida ao Curso de Bacharelado em Sistemas de Informação como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientadora: Prof. Ivenilton Alexandre de Souza Moura

PICOS - PI

2016

FICHA CATALOGRÁFICA

Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

E746p Escobar, Flávio dos Santos.

Protótipo modular do sistema de intercomunicação *arduino-raspberry PI* (ArPI) / Flávio dos Santos Escobar.– 2016.
CD-ROM : il.; 4 ¾ pol. (49f.)
Monografia (Curso Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.
Orientador(A): Prof. Esp. Ivenilton Alexandre de Sousa Moura

1. *Intercomunicação-Arduino*. 3. Sistemas Embarcados. I. *Raspberry*. Título.

CDD 005.4

PROTÓTIPO MODULAR DO SISTEMA DE INTERCOMUNICAÇÃO ARDUINO – RASPBERRY PI (ArPI)

FLAVIO DOS SANTOS ESCOBAR

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 19 de fevereiro de 2016

Profª. Esp. Ivenilton Alexandre de Souza Moura
Orientador

Frank César Lopes Véras
Prof. Me. Frank César Lopes Véras
Membro

Prof. Me. Fredison Muniz de Sousa
Membro

Dedico este trabalho a minha Família, em especial a minha mãe Iranilde Paes dos Santos que se fez presente em todos os meus dias me orientando e dando força. Aos meus irmãos que eu amo incondicionalmente e ao meu pai Raimundo Escobar, que será sempre lembrado com muito amor.

AGRADECIMENTOS

Quero agradecer a minha Família por estarem sempre ao meu lado em todos os momentos, me apoiando e me dando forças para que eu possa conquistar meus objetivos. Aos meus pais Claudio Roberto de Souza e Iranilde Paes dos Santos que sempre acreditaram e investiram na minha formação e de meus irmãos. Agradeço a minha mãe em especial, por sempre ter insistido no meu crescimento educacional e na minha vivencia acadêmica. Agradeço a minha avó Ermila Paes de Santana pelos ensinamentos e conselhos de vida que me fizeram crescer como pessoa. Por ter acreditado, acima de tudo, que a educação e o conhecimento são a riqueza que realmente importa.

Agradeço ao meu orientador Ivenilton Alexandre de Sousa Moura pelos ensinamentos e conhecimentos que me nortearam na construção desse projeto. Por me fazer olhar de diferentes perspectivas, por despertar em mim e em todos os seus alunos e alunas o senso critico e criativo. Agradeço por ter acreditado na minha capacidade de concluir esse projeto e também pela paciência e tempo dedicado nas orientações que nortearam minha pesquisa e desenvolvimento desse projeto. Agradeço aos meus professores por terem contribuído diretamente com a minha formação acadêmica. Especialmente as professoras Patricia Medyna e Alcilene Dalília por terem lutado a todo o momento pelo curso de Bacharelado de Sistemas de Informação-CSHNB trazendo melhorias e excelência para o mesmo.

Sou grato a todos os amigos que fiz durante minha vivência acadêmica. Aos meus colegas de curso, em especial aos meus amigos, Laurentino Junior, Alexandre Ramos e Luís Henrique pela amizade e experiências proporcionadas. Ao meu amigo Marcelo Damasceno por ter compartilhado comigo, além de grandes momentos, o seu talento confeccionando o logotipo do protótipo do sistema ArPI. E aos meus amigos Micael, Wesley, Lincoln e Wallace por terem compartilhado momentos de grande alegria e por sempre terem me apoiado nesta fase.

Agradeço a oportunidade que a academia me proporcionou, de conhecer diversas pessoas e dentre elas cultivar amizades valiosas. Amizades de pessoas incríveis como a do Erick, da Eveline, do Robson, do Cristiano do Mauricio. Quero agradecer a todos pelas experiências que compartilhei. Enfim, agradeço a todas as pessoas que contribuíram direta e indiretamente com a realização desta etapa importante da construção do meu futuro profissional e/ou acadêmico.

"A revolução libertará a sociedade
de seus flagelos e a ciência, o indivíduo".
(Galileu Gal)

A guerra do fim do mundo - Mario Vargas Llosa.

RESUMO

O presente trabalho visa à criação de um protótipo de um sistema de intercomunicação entre um microcontrolador *Raspberry PI model B+ Broadcom BCM2835* e, de um a, vários microcontroladores Arduinos (Atmega1280, Atmega328, entre outros modelos). Esse protótipo foi criado com o intuito de corroborar a hipótese da criação do sistema de intercomunicação (ArPI) e modularizar suas principais operações. O Objetivo principal foi reunir as operações fundamentais que este sistema possa vir a necessitar. Operações como: endereçar e listar dispositivos, trocar dados, fazer varreduras e gerenciar interrupções. O protótipo funciona em duas partes, uma embarcada no *Raspberry* PI (RPI) e a outra no(s) Arduino(s). As operações gerenciais que demandam mais custo computacional são embarcadas no RPI devido seu poder de processamento. Ao passo que nos Arduinos são embarcados os módulos de comunicação e módulos operacionais. O modulo de comunicação é inerente a todos os Arduinos a fim de permitir a comunicação entre os mesmos. Os módulos operacionais, por outro lado, variam entre os Arduinos dependendo das funções que os mesmos devem exercer. Esta ferramenta facilita, também, o gerenciamento de microcontroladoras Arduino, permitindo que projetos de automação possam trabalhar de forma conjunta e/ou coordenada.

Palavras-chave: Intercomunicação, *Arduino*, Sistemas Embarcados, *Raspberry PI*, microcontroladoras.

ABSTRACT

This work aims to create a prototype of an intercom system between a microcontroller Raspberry PI model B + Broadcom BCM2835 and one to several Arduinos microcontrollers (ATmega1280, Atmega328, among other models). This prototype was created in order to corroborate the hypothesis of the creation of the intercom system (ArPI) and modularize its main operations. The main objective was to bring together the fundamental operations that this system is likely to need. Operations such as: address and list devices, exchange data, making scans and manage interruptions. The prototype runs in two parts, one embedded in Raspberry PI (RPI) and the other (s) Arduino (s). The management operations that require more computational costs are embedded in the RPI due to its processing power. While the Arduinos communication modules and operating modules are embedded. The communication module is inherent in all Arduinos to allow communication there between. Operating modules, on the other hand, vary between Arduinos depending on the functions that they must perform. This tool facilitates also the Arduino microcontroladoras management, allowing automation projects can work together and / or coordinated way.

Keywords: Intercommunication, Arduino, Embedded Systems, Raspberry PI, microcontrollers.

LISTA DE FIGURAS

LISTA DE QUADROS

ABREVIATURAS, SIGLAS E CONVENÇÕES

RPI	RASPBERRYPI
I2C	INTER-INTEGRATED CIRCUIT
SPI	SERIAL PERIPHERAL INTERFACE
SDA	SERIAL DATA LINE
SCL	SERIAL CLOCK LINE
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
VCC	TENSÃO DE ALIMENTAÇÃO POSITIVA
GPIO	GENERAL PURPOSE INPUT/OUTPUT
GND	GROUND
PC	PERSONAL COMPUTER
CPU	CENTRAL PROCESSING UNIT
BIT	BINARY DIGIT
BYTE	BINARY TERM

SUMÁRIO

1 INTRODUÇÃO

A maior rede de comunicação que possuímos nos dias atuais é a *Internet*. Trata-se de uma rede mundial que conecta diversos dispositivos, permitindo a troca de dados e informações entre eles. Através dela pessoas do mundo inteiro podem se comunicar quase que instantaneamente. Basta possuir um computador, uma rede que os conecte, e que os dispositivos compartilhem os protocolos de comunicação responsáveis por controlar o processo. Esses dispositivos tornaram-se portas de acesso a maior e mais rápida rede de compartilhamento que se tem conhecimento.

Com a evolução tecnológica que ocorreu nas últimas décadas, os dispositivos tornaram-se cada vez menores e mais capazes. Assumindo diversas formas e propósitos com o objetivo de suprir necessidades que lhe foram demandadas ou apresentar soluções a problemas dos mais diversos tipos. A quantidade de dispositivos que um usuário comum passou a utilizar subiu consideravelmente nos últimos anos, com expectativa de cerca de 7 (sete) dispositivos interconectados por pessoa até o ano de 2020 (INTELDIALOGOTI, 2016).

Alguns destes dispositivos evoluíram em placas de circuito integrado, muito menores que os *notebooks* modernos. É o caso do *Raspberry* PI, um microcontrolador do tamanho de um cartão de crédito, porém capaz de realizar rotinas básicas quase da mesma que os computadores pessoais (PC) atuais. Rotinas como: conectar-se a *internet*, executar mídias, rodar aplicações, entre outros. Por outro lado, dispositivos integrados como o Arduino, por exemplo, são capazes de interagir com sensores e atuadores, tornando-os ideais em projetos microcontrolados.

A plataforma Arduino é uma das mais populares entre a comunidade de desenvolvedores de projetos microcontrolados. Com ela é possível automatizar pequenas atividades como, abrir e fechar portas, janelas e portões; ligar e desligar aparelhos e luzes; e automatizar sistemas como, o de irrigação, fornecimento e captura. Porém quando se deseja automatizar várias atividades é preciso utilizar vários Arduinos. Programar cada 1 (um) conforme seu propósito e gerencia-los individualmente.

O interessante é que dispositivos, que são utilizados em projeto microcontrolados, trabalhem dentro de uma rede de comunicação onde possam trocar informações de forma colaborativa. E que haja, assim como na *internet*, um sistema ou *framework* que coordene os

dados que trafegam na rede. Um sistema que possa acessar todos os Arduinos, por exemplo, fazendo pedidos e requisições, e controlando suas atividades. Este sistema facilitaria consideravelmente o gerenciamento de uma variedade substancial de microcontroladores deste tipo. Permitindo ao usuário além de tudo, manipula-los de um único ponto.

Devido à ampla diversidade de dispositivos que existe hoje no mercado o grande desafio fica por conta de trabalhar com esses dispositivos em conjunto. Isto porque elas são de plataformas diferentes e costumam operar em frequências diferentes, com voltagens diferentes, possuindo diferentes linguagens padrão. Essas diferenças dificultam a construção de projetos que utilizam plataformas distintas trabalhando em conjunto.

É possível encontrar na *internet* várias formas de comunicação entre microcontroladoras do mesmo arquétipo ou de arquiteturas diferentes. A maioria delas utilizando protocolo *Serial Peripheral Interface* (SPI), um tipo de barramento utilizado na comunicação de dispositivos eletrônicos. Toda via, este protocolo tornou-se impraticável em projetos que visam interconectar uma ampla variedade de dispositivos simultaneamente. Surgiu então à necessidade de uma nova ferramenta de comunicação que suprisse tais exigências.

Segundo a Philips Semiconductors (2014), por volta de 1992 surgiu a primeira versão de um novo protocolo, chamado de *Inter-Integrated Circuit* (I²C) ou *Inter IC*. Diferente do barramento SPI, o I²C utiliza duas linhas de comunicação denominadas *Serial Data Line* (SDA) e *Serial Clock Line* (SCL). Na linha SDA trafegam dados e endereços, enquanto que a linha SCL é utilizada para o controle das informações. No protocolo I²C o endereçamento se dá por *software* e no SPI o endereçamento é feito através de uma linha direta entre o mestre e o escravo.

A intenção do I²C é facilitar a comunicação digital entre dois ou mais componentes de forma prática e simples. Sua comunicação digital permite instanciar uma variedade maior de dispositivos. Visto que a maioria das placas de circuito integrado possuem poucas portas programáveis disponíveis para a comunicação. Essa característica em especial o torna altamente recomendável em projetos microcontrolados como o abordado nesse trabalho.

1.1 Objetivo

O objetivo principal foi desenvolver um protótipo de um sistema de intercomunicação que conecte sistemas embarcados, especificamente o *Raspberry PI* e vários Arduinos. A finalidade do projeto é reunir, em um protótipo, os principais módulos a serem utilizados neste sistema. Possibilitando, em trabalhos futuros, a construção de um sistema que interconecte estes dispositivos (Arduino e *Raspberry PI* – ArPI), permitindo um gerenciamento e controle melhor das microcontroladoras Arduino.

1.2 Organização do Trabalho

Este trabalho esta organizado em 6 (seis) capítulos. No cap. 2 encontra-se todas as definições e referencias teóricas que fundamentam o projeto, e que servirão de base para a compreensão adequada do trabalho como um todo. Alguns conceitos também serão abordados para facilitar o entendimento dos processos de criação envolvidos na concepção do protótipo do sistema de intercomunicação. Conceitos como: protocolos de comunicação, internet das coisas, interconexão, microcontroladoras e etc.

No cap. 3 serão abordados os métodos e as ferramentas utilizadas para o desenvolvimento e a conclusão do projeto. Bem como discorrer todos os processos de criação e a utilização dos mesmos na composição final do protótipo. Salientando os principais pontos de enfoque no que se diz respeito à intercomunicação dos dispositivos em questão – RPI e Arduino.

Para facilitar a compreensão do funcionamento do sistema, todos os resultados serão discutidos no capitulo 4. No capitulo 5 é apresentado algumas hipóteses e opiniões do autor acerca do projeto e de trabalhos futuros, concluindo as atividades aqui abordadas. O sexto capítulo contem todas as referencias bibliográficas utilizadas na realização deste projeto.

2 REFERENCIAL TEORICO

Esse capítulo aborda as definições e os conceitos básicos necessários para a compreensão do projeto, a fim de esclarecer sua construção, bem como os processos que o levaram a seu estado final. Conceitos como o de microcontroladoras, protocolos de comunicação, automação e sincronização.

2.1 Microcontroladoras

Assim como os computadores pessoais, os sistemas embarcados têm se tornado amplamente utilizado nos últimos anos, afetando cada vez mais pessoas. Desde terminais bancários de caixas eletrônicos a aparelhos eletrodomésticos. Diariamente as pessoas são beneficiadas pelos serviços que esse tipo de sistema oferece (SANTOS, 2000).

Marwedel (2003) destaca que sistemas embarcados podem ser definidos como sistemas eletrônicos de processamento de informação embutidos em um produto de forma transparente para o usuário. Tais sistemas são vistos todos os dias, nos roteadores, ou celulares, TVs, relógios, etc. As pesquisas nessa área são relevantes no contexto atual, principalmente entre as empresas do ramo eletrônico. A capacidade de controle desses sistemas poderia ser muito bem aproveitada caso houvesse um sistema eficaz de troca de dados. Contudo no que se refere a estas microcontroladoras (RPI e Arduino) é raro encontrar uma rede de comunicação inteligente capaz de suportar uma quantidade grande de dispositivos.

Com a explosão tecnológica precedida da criação da *internet* é possível perceber a necessidade, cada vez maior, de se conectar e conectar as coisas. Através de sensores microcontrolados é possível obter uma quantidade substancial de informações e dados processados na palma da mão. Inserido nesse contexto há uma gama de microcontroladoras capazes de recolher, analisar, gerenciar e processar informações. Dentre essas, algumas se destacam em determinadas atividades.

O Arduino, por exemplo, é umas das plataformas mais utilizadas em projetos microcontrolados, com prerrogativas que vão desde o seu custo, até a facilidade de trabalhar com tal dispositivo. O *Raspberry* PI por sua vez, vem conquistando seu espaço nesse meio

por características que o fazem excepcional, sobretudo pela sua vasta comunidade de colaboradores.

> O Arduino foi projetado especificamente para projetos de design interativa e programação de microcontroladores. Ele é uma plataforma de prototipagem eletrônica. Já o Raspberry é um computador completo, de baixíssimo custo. O objetivo dos dois é semelhante: incentivar o ensino de habilidades relacionadas à computação e eletrônica a preços baixos, mas os dois servem a funções distintas.(Lemos, 2014)

2.1.1 Raspberry PI

O *Raspberry* PI (RPI) é uma placa de circuito integrado com tamanho aproximado de um cartão de crédito e de baixo custo. Idealizada por um grupo de estudantes ligados ao Laboratório de Informática da Universidade de Cambridge, o RPI é um dispositivo que permite o desenvolvimento de projetos dos mais diversos tipos. Que vão desde a criação de projetos integrados e de automação, através de seus pinos programáveis, a projetos de maior abstração (*RASPBERRYPI, 2016*).

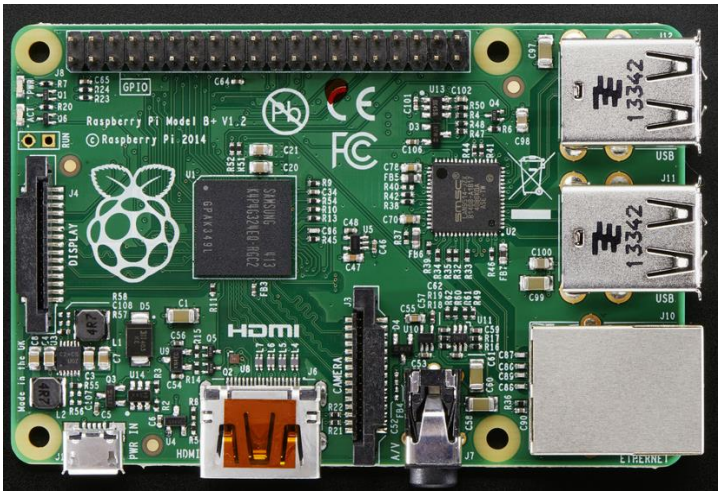

Figura 1 - Raspberry PI model b+. Fonte (https://www.adafruit.com/images/970x728/1914-04.jpg).

O *Raspberry* PI ou RPI possui características de hardware que nem todas as outras microcontroladoras conhecidas possuem. O modelo b+ dispõe de 4 (quatro) portas USB, 1 (uma) porta HDMI, placa de rede *on-board*, *Ethernet* 10/100 RJ45, espaço para um cartão Micro SD, melhor eficiência energética em relação as versões anteriores. Possui também 40 (quarenta) pinos programáveis de entrada e saída (GPIO) responsáveis por prover uma interface entre as placas e os periféricos. Conta também com um processador de 700 MHz e

uma memória RAM de 512MB. Vantagens pelas quais este dispositivo foi escolhido para rodar a base do sistema (ADAFRUIT, 2016).

Dentre os sistemas operacionais que o RPI suporta, o Raspbian é um dos mais utilizados. Isso se deve a uma série e fatores, que vão desde a sua facilidade de instalação, até sua intuitividade. Open Source e baseado em Debian, o Raspbian, foi desenvolvido com o intuito de otimizar ao máximo os processos executados no *Raspberry* PI. Suporta também programas de desenvolvimento dos mais diversos tipos e possui cerca de 35mil pacotes pré-compilados de *software* em diversas linguagens (RASPBIAN, 2016).

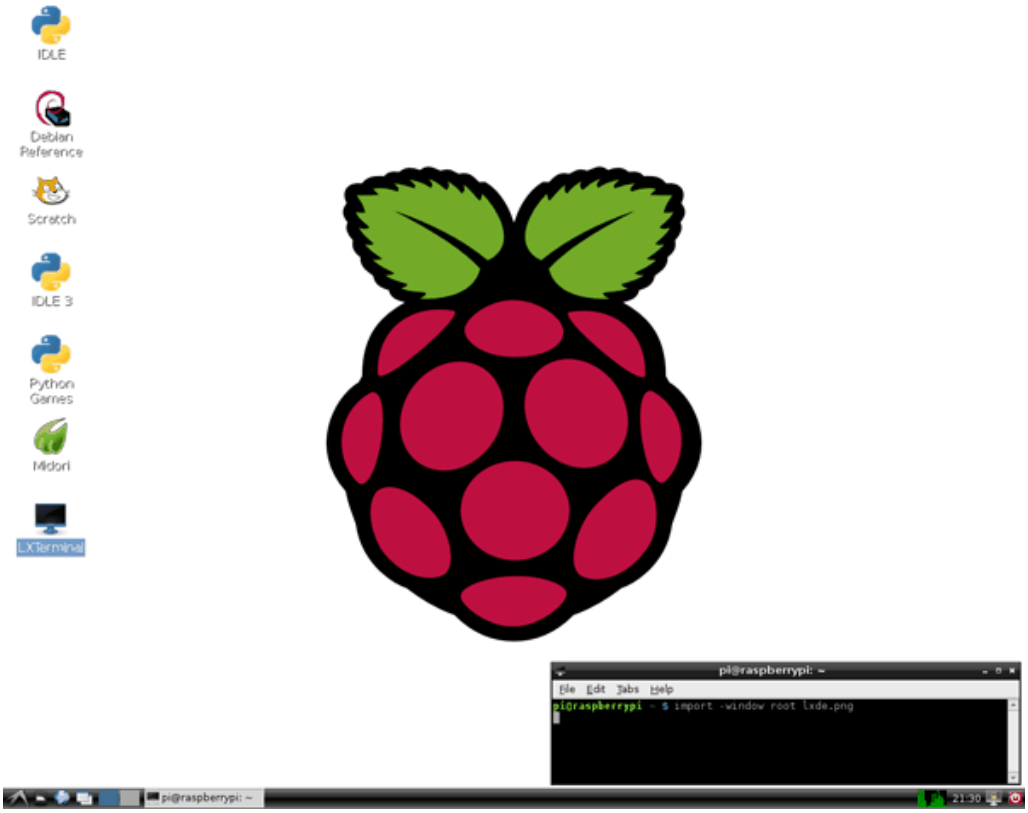

Figura 2 - Interface gráfica do Raspbian. Fonte (http://www.embarcados.com.br/wp-content/uploads/2014/07/lxde.png).

2.1.2 Arduino

A plataforma Arduino é uma das mais utilizadas no ramo, contendo uma variedade incrível de versões das quais se pode trabalhar, e com suporte a uma grande quantidade de sensores, periféricos, manipuladores e atuadores. Com sua politica de *"prototipagem de código aberto baseado em hardware easy-to-use"*, o Arduino se tornou uma das ferramentas mais simples, de fácil acesso e uso, e com uma das maiores comunidades desenvolvedoras da área (ARDUINO, 2016).

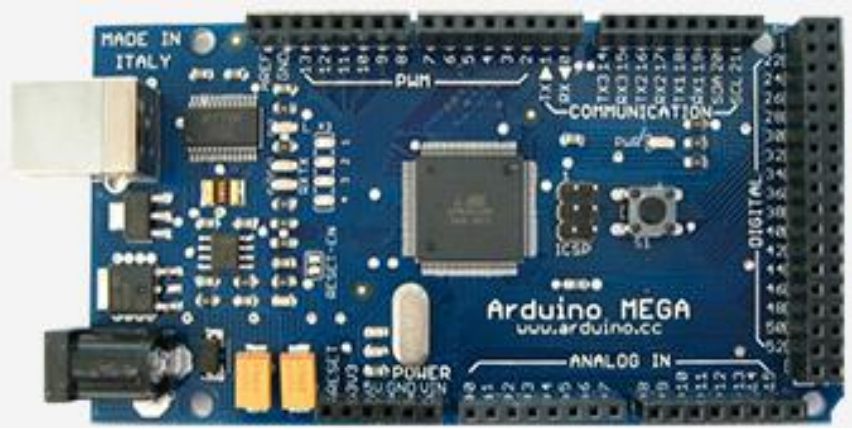

Figura 3 - Arduino Mega. Fonte (https://www.arduino.cc/en/uploads/Main/ArduinoMega.jpg).

Diferente do *Raspberry* PI, todo o contato do usuário com o *software* da micro controladora se resume a uma *Integrated Development Environment* (IDE). Este ambiente de desenvolvimento (IDE) é o ponto de partida para a construção de qualquer software que o usuário queira rodar na placa. Outras vantagens que tornam essa pequena placa em uma grande ferramenta de prototipagem são características como: o seu baixo custo, sua capacidade de operar em quase todas as plataformas usuais, totalmente *open source*, hardware e software extensíveis; e segundo sua própria comunidade possui um dos ambientes de programação, mas simples e intuitivos (ARDUINO, 2016).

Figura 4 - Ambiente de programação do Arduino. Fonte (http://www.natalnet.br/ura/wp-content/uploads/2013/10/arduino-ide.png).

2.2 Protocolos de comunicação

Projetos que têm a necessidade de utilizar um ou mais dispositivos de entrada e saída, e que, além disso, precisem trocar informações entre os mesmos, fazem uso de algum tipo protocolo de comunicação. Os protocolos de comunicação são responsáveis por trocar dados analógicos e/ou digitais entre dispositivos eletrônicos. E através deles, é possível interconectar sistemas integrados, trocar informações, atribuir dados e manipular entradas e saídas.

Existem vários tipos de protocolos de comunicação que são utilizados em projetos microcontrolados. Eles podem ser divididos em dois tipos: paralelos e seriais. A diferença fundamental entre eles esta na forma como são enviados os dados. Na comunicação em paralelo os dados são enviados simultaneamente por canais distintos. Ao passo que na comunicação em série os dados são enviados individualmente pelo mesmo canal.

Os protocolos de comunicação serial se dividem em síncrona e assíncrona. Os protocolos que entram na categoria de comunicação serial síncrona fazem uso de um sinal de sincronismo que permite a troca mais eficiente de dados e com menos erros na transmissão. A

comunicação assíncrona por outro lado transmite os dados sem o sinal de sincronismo, o tempo da transmissão deve ser previamente conhecido pelos dispositivos (AMADEU, 2012).

Os equipamentos eletrônicos têm agregado cada vez mais funções e recursos, tornando comum a utilização de uma variedade maior de circuitos integrados. Porém Sacco (2014) afirma que conectar tais circuitos através de barramentos de comunicação em paralelo tornaria as placas de circuito impresso muito caras e praticamente inviáveis. Contudo a comunicação serial permite que esses circuitos integrados se comuniquem como o próprio nome sugere, de forma serial.

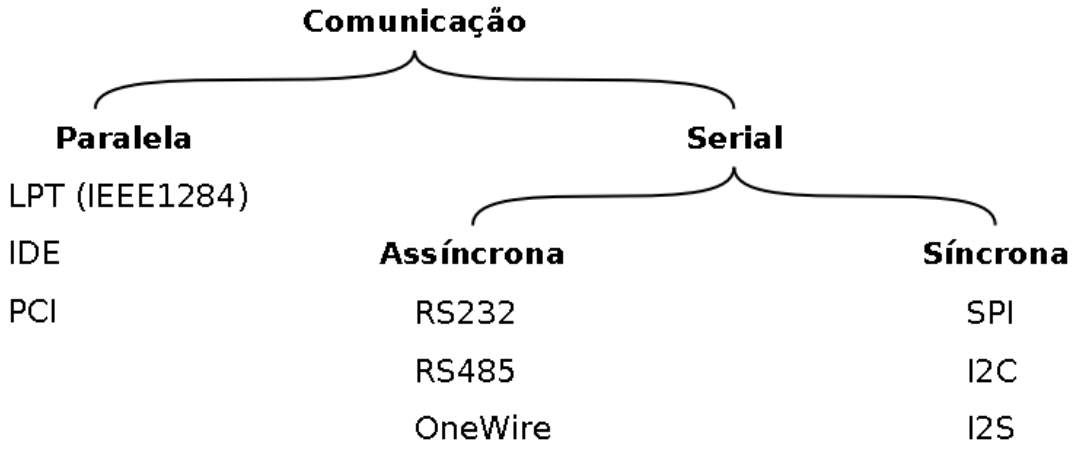

Figura 5 - Tipos de protocolos de comunicação. Fonte (http://www.embarcados.com.br/wp-content/uploads/2014/04/Tipos_de_comunicacao.png).

Um dos conceitos que se deve ter em mente quando se trabalha com comunicação serial síncrona, é a de mestre-escravo. A comunicação serial síncrona parte do pressuposto que um dispositivo (mestre) gera o sinal de sincronismo e um ou mais dispositivos (escravos) recebe (SACCO, 2014). Este conceito de mestre e escravo varia conforme a necessidade do projeto. Podemos ter sistemas de comunicação funcionando com 1 (um) mestre e vários escravos ou um sistema multimestres, por exemplo. É importante salientar que este conceito serve apenas para facilitar a compreensão de como a comunicação vai acontecer, uma vez que na comunicação serial síncrona algum sinal de sincronismo deverá ser gerado – normalmente o/um mestre.

Os protocolos de comunicação do tipo serial síncronos mais utilizados em placas de circuito integrado são os populares *Serial Peripheral Interface* (SPI) e a *Inter-Integrated*

Circuit (I2C). São amplamente utilizadas em projetos embarcados de baixo nível, fornecendo uma *interface* de comunicação bem simples e usual. As bibliotecas desenvolvidas em plataformas populares como o Arduino, o *Raspberry* PI, o *BleagleBone* são um dos principais motivos pela qual este barramento é bastante utilizado. Isto é, em microcontroladores como as citadas anteriormente que trabalham com periféricos dos mais diversos tipos, protocolos como esses se tornam indispensáveis.

2.2.1 SPI

O protocolo de comunicação *Serial Peripheral Interface* (SPI) foi desenvolvido originalmente pela Motorola com o intuito de fornecer uma *interface* de comunicação de curta distância. Utilizado principalmente em circuitos integrados este barramento se destaca principalmente pela sua simplicidade. Tornou-se popular quando foi apresentada pela Motorola no microprocessador Motorola 68000 e desde então foi adotado pela maioria dos fabricantes (BALDASSARI, 2016).

O SPI se baseia em comunicação serial síncrona, ou seja, há a necessidade que algum dispositivo gere o sinal de sincronia. O dispositivo que gera este sinal recebe o nome de mestre, os dispositivos que recebem o sinal são chamados de escravos. Este sinal, chamado de *clock*, é responsável por sincronizar a comunicação de tal forma que se adeque as necessidades do gerador do sinal (SACCO, 2014).

Para que haja a comunicação entre os dispositivos que utilizam esse protocolo, 4 canais/barramentos são necessários para estabelecer a comunicação. Um deles é disponibilizado para o *clock*, que partirá do mestre para os escravos sincronizando a comunicação. Outros dois são destinados à transferência de dados, sendo que um canal sempre transmite do mestre para o escravo **MOSI** (*Master Out Slave In*) e outro do escravo para o mestre **MISO** (*Master In Slave Out*). O canal restante é conhecido como **SS** (*Slave Select*), para cada escravo que se queira conectar, um canal **SS** deve ser disponibilizado. O *Slave Select* trabalha definindo o endereço de cada escravo, para que o mestre saiba para quem enviar e de quem receber. Contudo não há bit de verificação, ou seja, o mestre pode estar enviando para nenhum dispositivo e não ter conhecimento disso (Sacco, 2016).

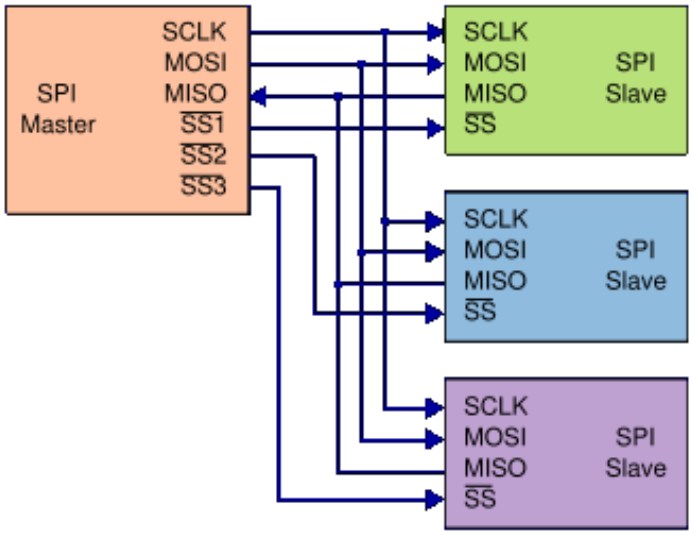

Figura 6 - Canais de comunicação do barramento SPI. Fonte (http://s3.amazonaws.com/magoo/ABAAAe7xoAL-8.jpg).

Segundo Sacco (2014), utilizando esse protocolo é possível atingir taxas de transferência de cerca de 1Mbps, o que é uma ótima taxa em projeto integrados. Porém quando o objetivo é conectar uma variedade considerável de dispositivos este protocolo se torna infactível. Isto é devido à necessidade de ter que disponibilizar um canal (pino) da placa para cara escravo; o *Raspberry* PI modelo b+, por exemplo, possui 26 pinos programáveis (GPIO). Este gargalo é o principal motivo pela qual este protocolo é deixado de lado em projeto de exigem grandes quantidade de dispositivos interconectados.

2.2.2 I2C

O barramento I2C ou *Inter-Integreted Circuit* é um protocolo de comunicação desenvolvido pela *Philips Semiconductors* seguindo o modelo serial síncrono. Assim como o SPI, este protocolo é amplamente utilizado em projetos que visam conectar dispositivos de circuito integrado. Uma característica desse protocolo é a sua versatilidade, isto é, a capacidade de trabalhar com diferentes tipos de tecnologias sem que haja incompatibilidade ou conflitos na comunicação (BYTE PARADIGM, 2016).

Diferente do SPI, este protocolo necessita de duas linhas/canais de comunicação e tem a capacidade de interconectar até 127 dispositivos. Uma dessas linhas é chamada de *Serial Data Line* (SDA), por onde trafegam os dados e endereços. A outra linha é chamada de *Serial Clock Line* (SCL) que assim como no SPI, é responsável por sincronizar a comunicação entre

o mestre e os escravos. A linha SDA possui a característica de ser bidirecional, isto permite que os dados possam ser enviados e recebidos através da mesma via, contudo somente um dispositivo pode utilizar o canal por vez (SACCO, 2014).

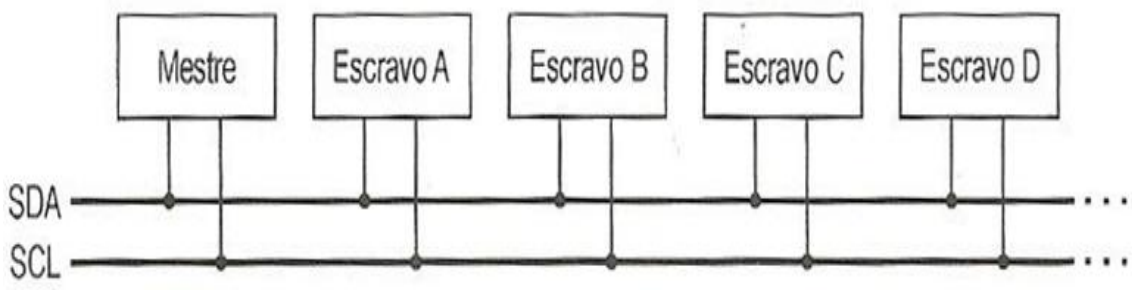

Figura 7 - Linhas de comunicação do barramento I²C. Fonte (Marciel, Braian K. Barramento Serial I²C e SPI).

Outra característica pertinente desse protocolo é a forma como é feito o endereçamento dos dispositivos escravos. Enquanto no SPI existe uma linha (SS) que vai do mestre até o escravo delimitando fisicamente a rota que os dados devem seguir, no I2C o endereçamento é feito por software. Quando é preciso fazer uma transmissão o mestre envia uma série de bits em grupos de 8 (oito), sendo que os 7 (sete) primeiros bits definem o endereço do escravo, e o oitavo bit define a operação (leitura ou escrita). A partir dai o mestre sabe para qual dispositivo o fluxo dos dados deve seguir (BALDASSARI, 2016). O método utilizado para endereçar os dispositivos acaba por definir a quantidade máxima de endereços que podem ser utilizados, uma vez que dispondo de 7 (sete) bits para endereçar só é possível gerar 128 possíveis endereços. O endereço 0x00 é reservado para uso exclusivo do protocolo, restando os 127 endereços que podem ser utilizados dentro do protocolo (SACCO, 2014).

Uma vez que o protocolo de comunicação I2C faz não faz o uso de um canal especifico que selecione o escravo e define a rota pela qual o fluxo de dados deve seguir alguns métodos devem ser aplicados. Estes métodos são responsáveis por coordenar os processos necessários para que ocorra a transmissão dos dados. Essa particularidade do I2C é solucionada por um conjunto de processos que define qual é o endereço, qual processo deve ser executado e quais os dados que deverão ser transmitidos.

Outra característica consequente do não uso desse canal (SS) que seleciona o escravo é tornar a comunicação *half-duplex*. Comunicações deste tipo são por definição vias de mão dupla que só permitem que o canal seja utilizado por um dispositivo por vez. Isto quer dizer

que é possível enviar e receber informações pelo canal de comunicação, porém só é possível ou enviar ou receber, nunca os dois ao mesmo tempo.

2.2.2.1 Transmissão

A priori, quando é preciso realizar uma transmissão, uma condição de *Start* é enviada sinalizando que a comunicação vai começar a partir dali. Essa condição é ativa quando o nível lógico do SDA muda de alto (*pull up*) para baixo (*pull down*) enquanto o canal SCL estiver em nível lógico alto. Quando essa condição é ativa o barramento deve ser considerado ocupado, pois há uma transmissão em andamento (BALDASSARI, 2016).

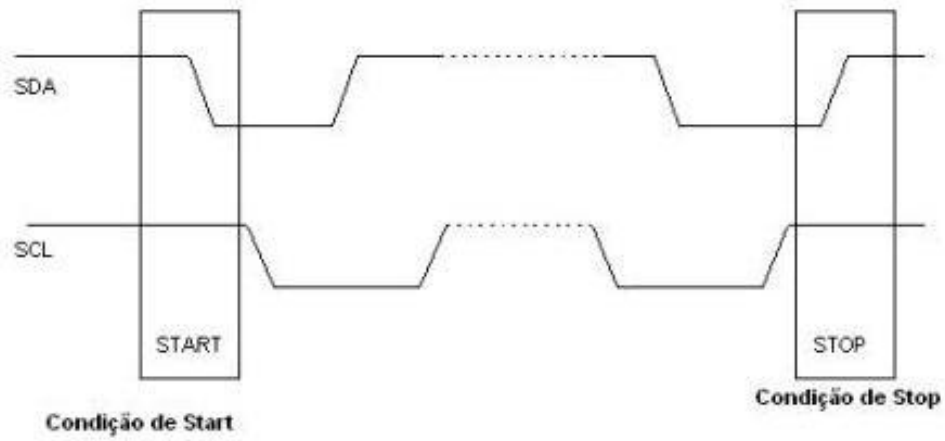

Figura 8 - Condições de Start e Stop do protocolo I2C. Fonte (http://1.bp.blogspot.com/-KX05jG5ComI/UL9JqSixZTI/AAAAAAAAA2Y/3Rsj6-ZMPT0/s640/I2C_02.jpg).

Logo após a condição de *Start* o mestre ativa o *Serial Clock Line* (SCL). A partir dai os bits são enviados um a um pela *Serial Data Line* (SDA), devendo cada bit assumir o nível lógico correto enquanto o *clock* mantém o nível lógico baixo. O dispositivo escravo deve ler os bits enquanto o nível lógico do *clock* for alto, nesse momento o bit deve se manter estável na linha SDA até que tenha sido entregue (SACCO, 2016).

A transmissão é encerrada quando uma condição de parada é detectada. Quando o nível lógico da SDA sai de baixo para alto enquanto o *clock* se mantem no *pull up* a condição de *Stop* é determinada. Esta condição finaliza a transmissão de dados uma fração de segundos depois de detectada liberando o canal para ser utilizado novamente.

3 PROTÓTIPO MODULAR DO SISTEMA DE INTERCOMUNICAÇÃO ARDUINO-RASPBERRY(ArPI)

Este projeto foi idealizado como parte de um sistema de intercomunicação que tinha o intuito de comunicar uma rede de sensores e atuadores controlados por métodos de inteligência computacional. O sistema residência como um todo deveria aplicar determinados controles inteligentes em uma moradia. No mais ele deveria ser capaz de interagir e aprender com seus moradores. A plataforma Arduino, seria utilizada para automatizar determinadas atividades cotidianas deixando os controles inteligentes por conta do *Raspberry*.

O projeto da residência foi dividido em três etapas de desenvolvimento. Na primeira etapa seriam definidas as atividades a automatizar e escolher as plataformas que pudessem ser utilizadas para essa tarefa. A segunda etapa consistia na implementação de mecanismos de inteligência responsáveis por reconhecer padrões e determinar o aprendizado do sistema a fim de melhorar a interação do usuário com o mesmo. O ultimo grande desafio consistia em criar um sistema que permitisse a comunicação entre estes dispositivos, uma vez que eles trabalhariam em conjunto.

Em razão da quantidade de avaliações e pesquisas acerca de qual dispositivo utilizar para automatizar as atividades residenciais como: ligar e desligar aparelhos sob comando ou condição de presença do morador(a), detectar os moradores por voz ou reconhecimento facial, reconhecer focos de perigo (incêndio, inundação, assalto), abrir e fechar portas e portões, entre outras atividades, a plataforma Arduino foi a que mais se destacou. Este microcontrolador possui um bom suporte ao uso de sensores e atuadores e possui uma vasta comunidade de colaboradores. É possível encontrar, no site oficial, diversos tutorias, exemplos e projetos relacionados que podem ajudaram muito do decorrer do trabalho.

Os mecanismos inteligentes, por outro lado, não poderiam ser embarcada no Arduino devido à quantidade de processamento e memória necessários para rodar a aplicação. Sabendo disso a alternativa foi utilizar a placa de circuito integrado *Raspberry* PI devido a sua capacidade de processamento e memória. Por meio dela é possível rodar todas as aplicações, contando ainda com características extras que podem ser utilizadas em trabalhos futuros.

O grande problema de trabalhar com diferentes plataformas -em conjunto- é fazê-las se comunicarem. Existem protocolos de comunicação como *o Inter-Integrated Circuit* (I2C)

ou o *Serial Peripheral Interface* (SPI) que determinam os processos responsáveis por garantir a transmissão, o endereçamento e as condições para que ocorra a comunicação. Porém esses protocolos operam como ferramentas para estabelecer a comunicação, todo o processo de enviar, recolher e tratar os dados ainda deve ser feito pelo desenvolvedor por meio de bibliotecas especificas do barramento.

No decorrer do projeto constatou-se que dispor de um sistema de compartilhamento de informações entre essas duas plataformas não só permitiria que o sistema funcionasse, mas também abriria um leque de possibilidades para projetos que pretendem trabalhar com essas placas em conjunto. A partir dai foi dado inicio a pesquisa que tinha por objetivo a construção um protótipo que modulariza-se as principais atividades do sistema de intercomunicação (ArPI). O sistema deveria trabalhar como um *framework* rodando em duas partes:

- A primeira parte seria embarcada no *Raspberry* PI (Mestre) responsável por gerenciar os Arduinos (Escravos) e trocar informações entre os mesmos.

- A segunda parte seria embarcada nos Arduinos; aqui o sistema funcionaria dependendo das atividades a qual o Arduino seria incumbido, isto é, há um módulo genérico necessário à comunicação –que todos os Arduinos devem possuir- e os módulos independentes que variam conforme a atividade varia. A idéia é que fosse possível que usuários do sistema pudessem criar seus próprios módulos de forma colaborativa tornando o sistema cada vez mais abrangente.

No durante as pesquisas iniciadas no pré-projeto do presente trabalho, foi possível observar que algumas operações seriam essenciais na construção do sistema de intercomunicação (ArPI). Por este motivo o enfoque deste projeto foi construir um protótipo que reunisse tais operações a fim de constatar a possibilidade da construção do sistema. Neste protótipo é possível listar os dispositivos em uso, trocar dados entre os mesmos, reendereça-los e atualizar a lista de dispositivos conectados na rede. Essas operações servem de ferramentas na construção e desenvolvimento de outras atividades dentro do sistema. O protótipo ainda cumpre com um papel importantíssimo na realização dos testes que envolvem a rede de comunicação entre os dispositivos em questão.

Durante esse capitulo será discorrida à problemática geral da criação do protótipo, bem como a concepção dos módulos de comunicação. Também serão abordados aspectos do protótipo como, o tratamento das informações, a *interface* e controle de interrupção.

3.1 Problemática

O grande desafio de se conectar dispositivos distintos e que, na maioria das vezes foi desenvolvido para trabalhar só ou em frequências diferentes, é justamente o fato de ter que lidar com conceitos diversos e fazê-las trabalhar em conjunto. Muitas vezes os dispositivos não compartilham da mesma frequência de operação, ou trabalham em voltagens diferentes. Então criar métodos para solucionar esses problemas é fundamental na construção do protótipo do sistema de intercomunicação (ArPI).

Sabendo disso fica claro deduzir que o maior desafio no desenvolvimento desse projeto foi ter que trabalhar com arquiteturas diferentes. Elas operam com voltagens e frequências distintas e não compartilham, nem mesmo, da linguagem de programação, sobretudo elas deveriam trabalhar em conjunto. O *Raspberry* PI, por exemplo, opera com voltagem de 3.3v, mesmo possuindo um pino programável (GPIO) que fornece 5v. Todos os outros pinos programáveis trabalham com voltagens de 3.3v. Esta informação torna-se pertinente uma vez que se sabe que o Arduino opera com voltagens de 5v, e no caso de interligas sem tomar as devidas precauções poderia facilmente danificar o RPI.

As características distintas de cada dispositivo que se tornaram grandes problemas na comunicação são:

- As linguagens de programação: o Arduino é uma plataforma de programação que possui suas principais bibliotecas baseadas em C e C++, incluindo seu compilador. Essa linguagem é amplamente utilizada desde a década de 70, e altamente recomendada em projetos de baixo nível como o apresentado neste trabalho. O *Raspberry* PI por outro lado foi criado com o intuito de trabalhar com linguagens de programação de alto nível como o Python, mesmo possuindo suporte a linguagens de médio a baixo nível. As bibliotecas necessárias para fazer o uso do protocolo de comunicação I2C estão mais facilmente disponíveis na IDLE do Python e possuem melhor suporte, basta fazer o *download* delas na *internet* através do site oficial. Trabalhar com duas linguagens tão diferentes quanto C++ e Python demanda bastante

cautela devido à mudança constante na logica de programação que o desenvolvedor se sujeita no decorrer do projeto. Deve-se tomar bastante cuidado em organizar os parâmetros de entrada e saída que eles compartilham.

- Voltagens diferentes: mesmo que ambas possuam pinos VCC que fornece 3.3v e 5v não faz diferença. A diferença crucial está na voltagem de operação dos pinos programáveis. Isto é, o Arduino fornece 3.3v na VCC, porém a voltagem padrão utilizada nas portas programáveis, responsáveis pelo sistema de interrupção, operam com 5v. Uma vez que o controle de interrupção faz o uso de um canal compartilhado que opera em *pull up* (nível logico alto) é preciso que haja uma conversão do nível lógico entre os dispositivos. Alguns dispositivos, como um conversor logico bidirecional, cumprem esse papel.

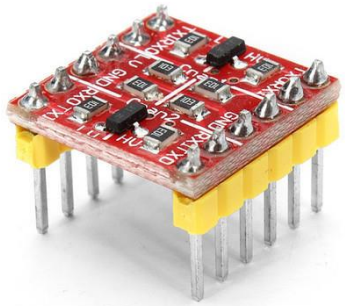

Figura 9 - Conversor de nível lógico 3.3v-5v bidirecional. Fonte (http://s3.amazonaws.com/img.iluria.com/product/14770B/4E587A/450xN.jpg).

- Sincronia: quando se troca informações entre o RPI e o Arduino é preciso calcular o tempo de resposta que cada microcontrolador possui. Sempre que um dispositivo envia um sinal, é necessário que o canal de comunicação esteja livre caso contrário haverá conflitos na transferência de pacotes e pode causar até queda na comunicação. Durante o desenvolvimento do protótipo do sistema de comunicação, medir e estimar o *delay* de cada dispositivo se tornou um trabalho necessário.

3.2 Interface

A principal forma como os usuários interagem com os *softwares* é por meio de uma *interface*. Elas são responsáveis por facilitar o modo como os usuários interagem com os sistemas. A interação dos usuários com os softwares estão cada vez mais transparentes, e têm

se tornado tão intuitivos que às vezes nem se percebe que estão ali. Contudo durante o desenvolvimento do projeto a *interface* assumiu outro papel.

No decorrer da construção do protótipo do sistema de comunicação os processos que eram desenvolvidos precisavam ser testados. Visto isso, foi criada uma *interface* que tinha por objetivo fornecer entradas e saídas da forma mais simples possível, a fim de que os resultados fossem testados com facilidade. Isto permitiu certa agilidade nos processos de desenvolvimento do protótipo.

Um pequeno *menu* foi construído contendo os principais módulos necessários a comunicação entre os dispositivos. Por meio dele é possível atualizar a lista de dispositivos conectados a rede, conferir quais os endereços estão sendo utilizados, enviar um dado (para fins de teste foi escolhido um numero inteiro de 1 a 255) para que o Arduino armazene, recuperar um dado de qualquer dispositivo em rede, reendereçar um dispositivo. O *menu* foi criado por comando da linguagem Python que escreve no terminal do Raspbian tendo seu retorno também pelo mesmo.

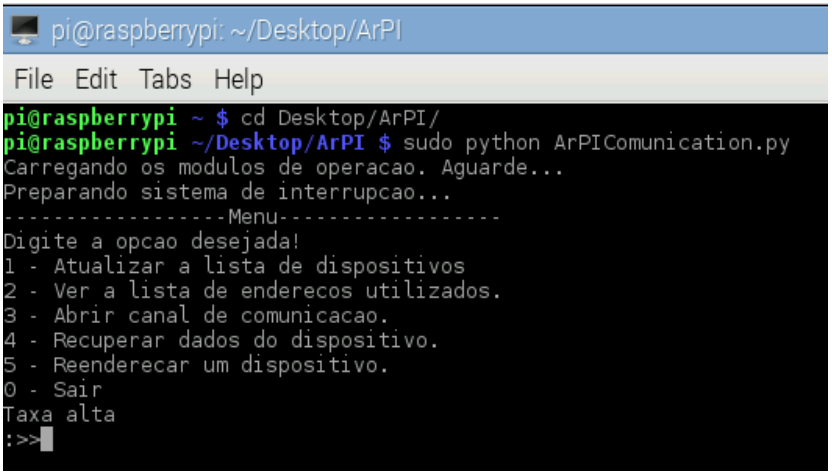

Figura 10 – *Menu* de acesso do protótipo ArPI. Fonte: Elaborada pelo autor (2016).

Outro fator pelo qual o acesso ao protótipo do sistema ArPI é feito pelo terminal do Raspbian é a necessidade do sistema de possuir permissões especiais. Assim como em outros sistemas baseados em Debian e Linux, o Raspbian possui politicas de segurança de acesso aos arquivos do sistema. Para criar, apagar ou modificar determinados arquivos dentro do sistema operacional é preciso ter permissão. Esta permissão é concedida ao usuário *root* do sistema operacional, isto é, ao usuário *raiz* que possui as permissões necessárias para alterar tudo dentro do mesmo.

Para salvar a lista de endereços dos dispositivos em rede o protótipo do sistema faz uso de um arquivo de texto. Este arquivo é criado pela primeira vez quando o sistema é iniciado, a partir de então ele é modificado à medida que os dispositivos são adicionados ou retirados da rede. Para que esta operação seja possível é necessário adquirir permissão de usuário *root* utilizando um comando que deve ser executado no terminal. Assim, antes de iniciar o sistema, a permissão é atribuída ao *script* como um todo.

3.3 Módulos de Comunicação

Os módulos de comunicação reúnem todos os processos responsáveis por estabelecer a comunicação entre o *Raspberry* PI e os Arduinos. A comunicação entre esses dispositivos funcionam respeitando determinadas condições. Essas condições coordenam o fluxo das informações entre os dispositivos para que não haja problemas como, conflitos, queda na comunicação, *delay* e ruídos.

A transmissão de informações acontece de forma quase que unilateral, isto é, o processo de comunicação sempre começa pelo *Raspberry* PI (mestre). Existem também alguns parâmetros necessários para que a comunicação possa ser efetuada. Esses parâmetros variam de acordo com a operação que irá ser executada.

Segue os passos que o RPI necessita realizar para armazenar uma informação num Arduino:

- O *Raspberry* PI executa um comando com os seguintes parâmetros: endereço do dispositivo e o dado correspondente à operação.

- Assim que o Arduino recebe o dado ele verifica se o dado é uma operação ou informação. Caso seja uma operação ele retorna o mesmo dado para afirmar ao RPI que recebeu o mesmo e se prepara para receber a informação.

- O RPI por sua vez, executa outro comando que também possui o endereço do dispositivo e a informação que o usuário deseja armazenar.

- O Arduino recebe a informação e armazena ela na sua memória (EEPROM) e então se prepara para receber uma operação.

Os passos que o RPI seque para recuperar um dado de algum escravo são basicamente os mesmo, a única diferença é que quando o Arduino recebe a operação que envia a informação para o RPI ele simplesmente a envia sem ter a necessidade de se preparar para receber uma informação. Porém quando o RPI necessita reendereçar um dispositivo ele passa por etapas distintas como:

- O RPI executa um comando que envia a operação para o Arduino.

- O Arduino recebe a operação e reenvia o mesmo (a fim de garantir ao RPI que recebeu a informação). A operação determina o modulo de reendereçamento que existe no Arduino e então o ativa.

- O RPI verifica se o novo endereço que o usuário deseja enviar está disponível e então envia a informação contendo o novo endereço.

- O Arduino recebe a informação e direciona para o modulo de reendereçamento atualizando o endereço atual.

As operações como, atualizar a lista de dispositivos e verificar a lista de dispositivos utilizados sequem basicamente a mesma logica que o RPI usa para recuperar um dado do Arduino. Quando o usuário atualiza a lista de dispositivos, o sistema faz uma varredura testando todos os endereços possíveis. Ele envia um bit de verificação e aqueles que respondem ao RPI são tidos como operantes. Quando se deseja verificar quais os endereços utilizados, não é necessário estabelecer nenhuma comunicação entre o RPI e os Arduinos. Todos os endereços são armazenados em uma lista num documento de texto no diretório **tmp** do dispositivo.

3.4 Tratamento de Informações e Operações

Quando se trabalha com comunicação de baixa velocidade, como o I2C, e que só é capaz de transferir 1 *byte* por vez, torna-se difícil trabalhar grandes fluxos de dados trafegando na rede. E este processo fica ainda mais lento quando uma característica dessa rede é ser, também, *half-duplex*, isto é, somente um dispositivo pode esta transferindo por vez.

Para contornar estes problemas algumas práticas foram adotadas. A preocupação inicial é não fazer o uso constante e intensivo da rede, sendo assim alguns módulos foram criados. Estes módulos interagem através de operações que funcionam como indicadores definindo o que deve ser feito a seguir. Essas operações foram definidas como números inteiros positivos que variam de 1 a 4. Sendo o 1 (um) utilizado para reendereçar um dispositivos, o 2 (dois) recuperar um dado armazenado em um escravo, o 3 (três) armazenar dados em um escravo e o 4 (quatro) é utilizado para controle de interrupção.

Os módulos existentes, tanto no *Raspberry* PI quanto nos Arduinos, descrevem processos básicos do protótipo dando liberdade criativa a quem o utiliza adicionar novos módulos. Estes módulos no RPI são responsáveis por executar comandos como: recuperar um dado, reendereçar um dispositivo, armazenar informações do mestre para o escravo, atualizar a rede e etc. No Arduino os módulos são tratados de forma diferente; há aqueles responsáveis pela comunicação com o RPI –esses são comuns a todos os escravos- e os responsáveis pelas ações que o Arduino pode exercer.

Nota-se que a maioria dos dados que trafegam na rede são operações e informações relevantes de baixo custo. Como ainda não foi implementado funções que exigem que grandes fluxos de dados trafeguem na rede o tratamento das informações tornou-se simples. Caso essas funções venham a ser feitas deve-se criar métodos que fracione a informação envie blocos de dados *byte* a *byte*.

3.5 Controle de Interrupção

Em sistemas multiprogramados a CPU está sempre sujeita a ter que compartilhar o seu tempo de processamento entre vários programas e dispositivos, internos e/ou externos. O processo de interrupção permite que a CPU trabalhe num esquema de paralelismo, dividindo seu tempo entre os atores envolvidos. Cada requerente faz um pedido, e a CPU respeitando algumas regras previamente estabelecidas, elabora uma lista de prioridades que decide como serão executados cada pedido. Esta técnica permite um melhor aproveitamento do tempo de processamento da CPU, evitando que ela fique ociosa quando poderia estar executando outra requisição.

O conceito definido anteriormente descreve mecanismos utilizados em processos de interrupção de *hardware*. Diferente desses mecanismos, o controle de interrupção

implementado neste projeto é feito por meio de *software*. Entretanto a essência do recurso é compartilhada por ambos os ambientes.

Já foi explorado neste artigo, no cap. 3.4 e no cap. 2.2.2, as limitações do I2C no quesito transferência mútua de dados. Este protocolo, como foi citado, caracteriza-se por ser *half-duplex* o que acaba por impossibilitar a transferência simultânea de dados (enviar e receber). Em razão deste aspecto o controle de interrupção não pôde ser feito pela via (SDA) utilizada para comunicação dos dispositivos. A causa obvia é que não seria possível utilizar este canal para fazer um pedido de interrupção se o mesmo estivesse sendo utilizado para comunicação. Devido a este fator uma alternativa teve quer ser elaborada a fim de contornar essa adversidade.

A solução foi disponibilizar um canal compartilhado para interrupções que pudesse ser acessado a qualquer momento pelos dispositivos em rede. Este canal liga o mestre (RPI) a todos os escravos (Arduino) permitindo que eles façam uso deste serviço sempre que necessário. Dois métodos foram usados para controlar os pedidos de interrupção entre os dispositivos, um foi embarcado no mestre (RPI) e o outro nos escravos (Arduinos). O método inserido no mestre lê constantemente o resultado de uma porta programável (GPIO) do *Raspberry* PI. O nível logico da porta é mantido no *pull up* da placa, isto é, o RPI mantém a voltagem máxima do canal constante. Quando acontece o *pull down*, ou seja, quando o nível lógico da porta passa de 3.3v - voltagem utilizada na GPIO do RPI- para 0v, uma interrupção é detectada.

É importante salientar que o nível logico da porta programável do RPI é mantido em alta para que não haja leituras falsas. Durante os testes foi notada a presença de leituras falsa quando o nível logico da porta se mantinha no *pull down*. Por este motivo mantém-se o nível logico alto na GPIO, entendendo como interrupção a queda da voltagem nessa porta.

O método que foi inserido em todos os escravos é responsável por gerar um pedido de interrupção. Este pedido é gerado quando um escravo necessita utilizar algum recurso que o mestre possui. Então ele acessa o canal de interrupções e baixa o nível logico para 0, aumentando-o em seguida. Com base nisso o escravo espera o mestre fazer a requisição e envia o seu pedido.

O pedido feito pelo escravo é direcionado pelo canal compartilhado até o mestre (RPI), que por sua vez o recebe, interrompe o processo atual e inicia processo de interrupção. O estado atual do sistema é salvo e então o mestre começar uma varredura para identificar qual dispositivo fez o pedido aproveitando para atualizar o estado atual da rede e verificar se outros dispositivos também possuem pedidos pendentes. Depois de encontrado, o dispositivo escravo envia ao mestre a operação que deseja executar. O mestre processa o pedido do dispositivo requerente e então retorna ao estado anterior à interrupção.

4 RESULTADOS E DISCUSSÕES

Este capítulo contém os resultados das funções do protótipo do sistema de intercomunicação ArPI detalhando-as. Serão apresentados também os resultados dos testes realizados durante a construção do projeto.

4.1 Primeiro contato

Diferente do Arduino, o *Raspberry* PI faz uso de um sistema operacional similar ao utilizado em computadores pessoais. O Raspbian é um sistema operacional baseado em Debian, é também *open source* e de fácil uso. Através dele o usuário tem acesso às ferramentas disponíveis do Raspberry.

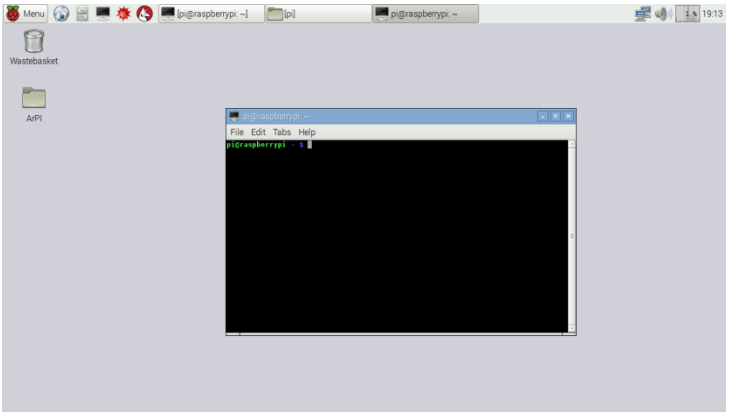

Figura 11 – Terminal do Raspbian. Fonte: Elaborada pelo autor (2016).

Como foi discorrido no capitulo anterior (Subcap. 3.2) o acesso ao protótipo do sistema é feito por meio do terminal do Raspbian. Para executar o protótipo do sistema ArPI é necessário que o usuário saiba o local que se encontra o *script* do mesmo, conforme a figura 12.

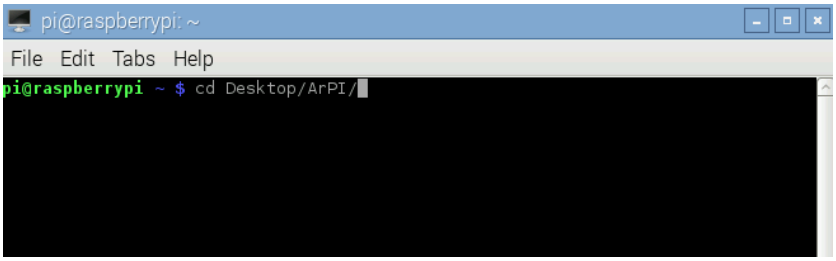

Figura 12 – Local do script do protótipo ArPI. Fonte: Elaborada pelo autor (2016).

Depois de localizado, deve-se inserir o local do *script* e então executar o seguinte comando: ‘sudo python **nomedoarquivo**.py’. Será requisitada a senha do usuário raiz (*root*) a fim de atribuir as permissões necessárias para executar o *script*. Depois de inserida a senha, o programa carrega alguns módulos e se inicia por completo.

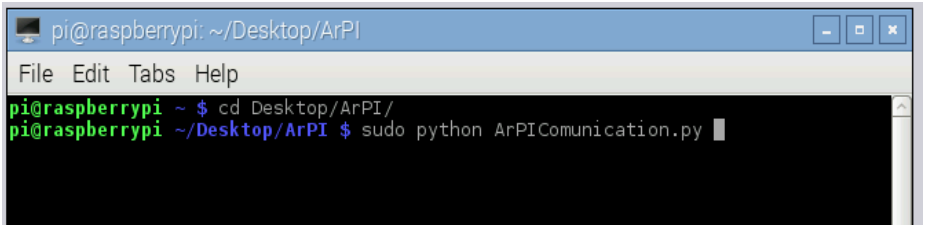

Figura 13 – Comando para executar o script do protótipo ArPI. Fonte: Elaborada pelo autor (2016).

4.2 Operações em funcionamento

As operações descritas aqui permitem que o usuário tenha acesso aos principais módulos do protótipo. Por meio do terminal do Raspbian o usuário tem acesso a atividades como: atualizar a lista de dispositivos, listar os endereços utilizados, abrir um canal de comunicação, recuperar um dado armazenado e reendereçar um dispositivo. Todas as atividades desenvolvidas neste protótipo confirma a possibilidade da concepção de um sistema que, utilizando esses módulos, integre operações adicionais. Operações como: gerenciar e monitorar os escravos de forma mais concisa, montar um sistema de fila de processos a fim de melhorar o sistema de interrupção, criar uma interface mais intuitiva e interativa.

4.2.1 Atualizar a lista de dispositivos

Toda vez que um uma varredura é feita pelo protótipo o custo computacional sobe consideravelmente durante aquele período. Isto se deve ao fato de muitas requisições serem feitas em sequencia, impedindo o uso da rede para outros fins. Em virtude disso a lista de dispositivos é salva pelo sistema em um documento de texto, para que toda vez que for preciso acessa-la, evitar-se-á varreduras desnecessárias. Este é um dos motivos que corroboram o uso de uma operação responsável por atualizar a lista dos dispositivos na rede; manter as varreduras sob controle.

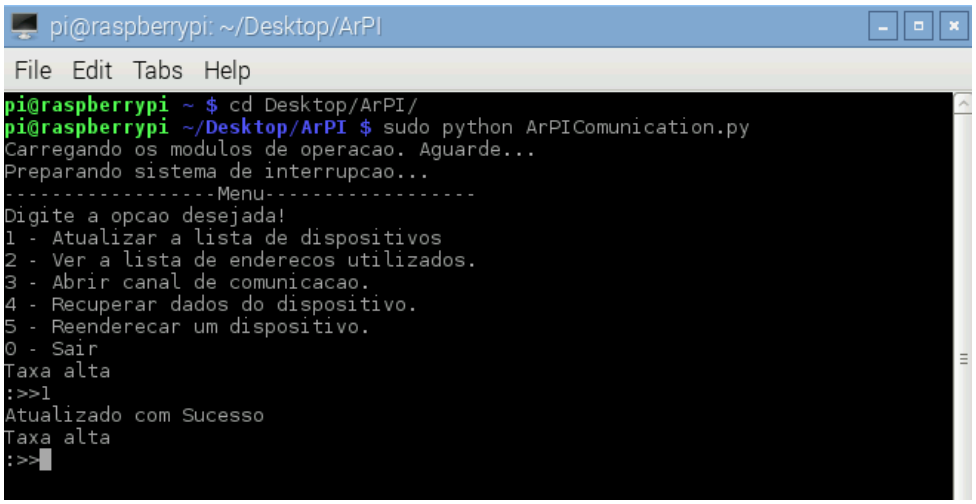

Figura 14 – Operação para Atualizar a lista de dispositivos. Fonte: elaborada pelo autor (2016).

Esta operação ativa um método dentro do protótipo do sistema chamada de '*loadModules()*' chama um outro método (atualizar()) responsável por fazer uma varredura na rede com o intuito de atualizar a lista de dispositivos conectados. Além disso, ela salva a lista de endereços recém-atualizada num arquivo de texto na pasta temporária do sistema operacional. Quando o algoritmo no RPI é iniciado pela primeira vez este método é ativado automaticamente a fim de prepará-lo para o uso imediato. O arquivo de texto é criado quando o programa inicia, se não houver dispositivos conectados na rede o arquivo salva 0 (zero) na lista. Dentro do modulo há um conversor que transforma a lista de inteiro para *string* para que seja possível salva-la, conforme a figura 15.

```
210 def loadModules():
211     #carrega a lista de enderecos e converte a lista de inteiro para string
212     lista_end = atualizar()
213     lista_str = []
214     i = 0
215 
216     while i < len(lista_end):
217         aux = lista_end[i]
218         #print type(aux)
219         cnv = str(aux)
220         lista_str.append(cnv)
221         #print type(cnv)
222         i = i + 1
223 
224     try:
225         arq = open('/tmp/listaEnd.txt', 'w')
226         arq.writelines(lista_str)
227         arq.close()
228     except:
229         pass
230 
231     time.sleep(1)
232     return -1
```

Figura 15 – Método *loadModules*(). Fonte: Elaborada pelo autor (2016).

4.2.2 Listar os dispositivos em uso

Quando esta operação é ativada um método chamado *readModules()* é executado. Este método recupera a lista de endereços do arquivo de texto salvo na pasta de arquivos temporários do Raspbian. Despois de recuperada a lista é convertida de *string* para inteiro, uma vez que os endereços são tratados como inteiro no resto do código fonte. Em seguida a lista é disponibilizada para o usuário por meio de uma impressão (retorno) na tela do terminal, conforme a figura 16.

```
234  def readModules():
235  #Faz a leitura da lista de enderecos e converte a lista de string para inteiro
236      try:
237          arq = open('/tmp/listaEnd.txt', 'r')
238          lista = arq.readlines()
239      except:
240          pass
241
242      i = 0
243      listar = []
244      while i < len(lista):
245          listar.append(int(lista[i]))
246          i = i + 1
247
248      return listar
```

Figura – 16 Método *readModules*(). Fonte: Elaborada pelo autor (2016).

Essa operação é um dos motivos pela qual o serviço de atualizar a lista de dispositivos, citado anteriormente, foi criado. Se todas as vezes que o usuário utilizasse este serviço fosse preciso fazer uma varredura na rede, a mesma se tornaria tão ocupada que a comunicação seria impraticável. É possivel observar na figura 17 que o processamento do *Raspberry* PI se mantém baixo como é recomendavel.

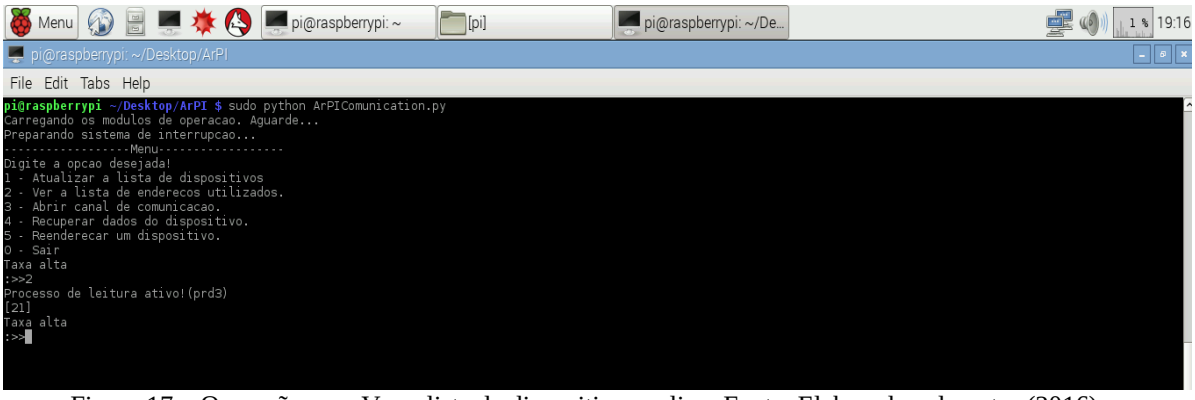

Figura 17 – Operação para Ver a lista de dispositivos online. Fonte: Elaborada pelo autor (2016).

4.2.3 Abrir um canal de comunicação

A idéia que deu origem ao inicio dessa pesquisa, até a criação do presente protótipo, foi interconectar o *Raspberry* PI com o(s) Arduino(s) a fim de possibilitar a gerencia e a troca de informações entre os mesmos. Para fins de teste foi inserida no protótipo uma operação responsável por abrir um canal de comunicação entre o mestre (RPI) e um escravo (Arduino). Este canal permite que o mestre armazene um dado na memória interna (EEPROM) do escravo para que possa resgata-lo posteriormente.

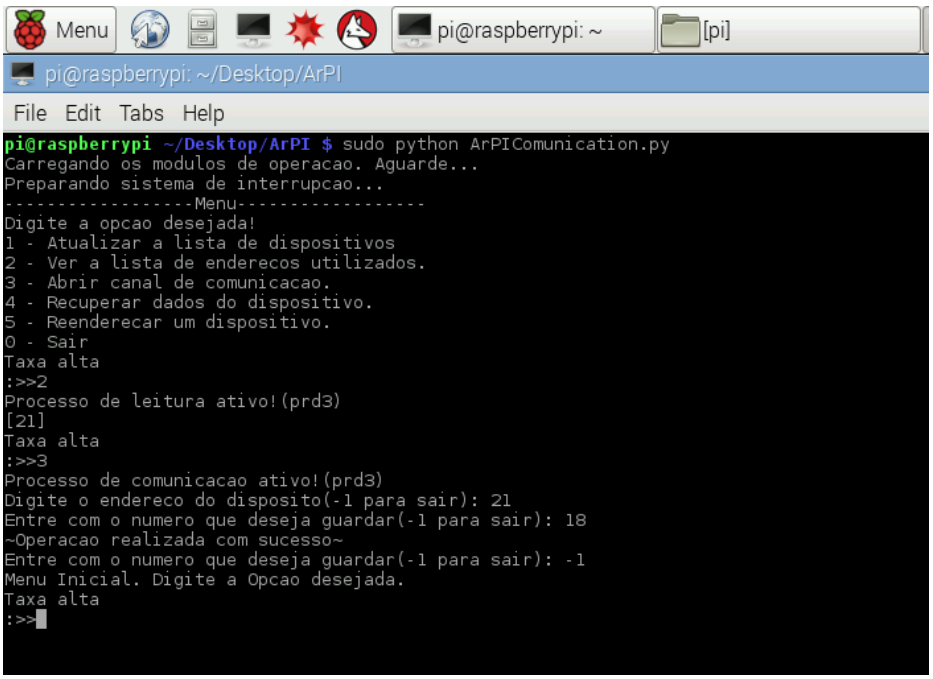

Figura 18 – Operação para Abrir um canal de comunicação. Fonte: Elaborada pelo autor (2016).

Como explorado no capitulo 3 do presente artigo o protocolo de comunicação I2C só permite que 1 *byte* seja transferido por vez. Neste protótipo nenhum método de fragmentar e enviar informações muito grandes foi implementado. Essa operação só permite enviar um numero inteiro positivo que varie de 1 a 254. Um fator que deve ser levado em consideração é a capacidade de memoria que os Arduinos possuem. Enviar grandes quantidades de informação pode superlotar a memoria dos Arduinos, acarretando em um mau funcionamento do sistema. Aumentar a quantidade de memoria dos Arduinos pode ser uma solução para utilizar este serviço nas próximas versões do protótipo. Contudo deve-se tomar cuidado em trabalhar com grandes volumes de dado em projetos embarcados, principalmente em projetos que utilizam barramentos de baixa velocidade (I2C, SPI).

O quadro a seguir demostra a esta operação em funcionamento. Uma série de dados é inserida, mesmo os que são maiores do que o suportado, e os resultados são observados no quadro 1.

Quadro 1 – Tabela de dados a serem armazenados em um escravo. Fonte: Elaborada pelo autor (2016).

Dado enviado pelo mestre (RPI)	Dado recebido pelo escravo (Arduino)	Dado Salvo pelo escravo (Arduino)	Resultado
8	8	8	Nº 8 Salvo com Sucesso
58	58	58	Nº 58 Salvo com Sucesso
108	108	108	Nº 108 Salvo com Sucesso
208	208	208	Nº 208 Salvo com Sucesso
256	Nenhum dado recebido	Nenhum dado salvo	Falha na operação
255	Nenhum dado recebido	Nenhum dado salvo	Falha na operação
1	1	1	Nº 1 Salvo com Sucesso

É possível observar que os dados enviados permanecem na faixa de 1 a 254. Os valores que não estão nessa faixa são tratados antes de serem enviados pelo mestre (RPI), por este motivo sequer são recebidos pelo escravo (Arduino), conforme é mostrado na figura 19.

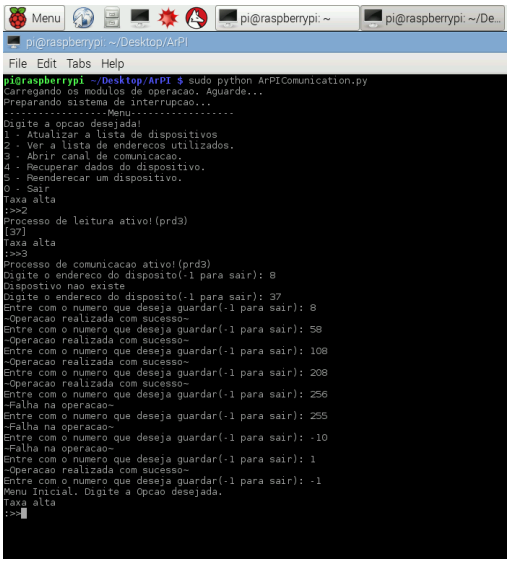

Figura 19 – Teste de inserção de dados a serem armazenados nos escravos. Fonte: Elaborada pelo autor (2016).

Os dados enviados pelo mestre (RPI) são recebidos pelo escravo e salvos em sua memoria interna (EEPROM), conforme a figura 20. Existe uma condição no modulo de reendereçamento dos escravos responsável por bloquear dados maiores que os que são enviados por vez. Esta condição foi inserida por segurança, visto que os dados já são tratados anteriormente.

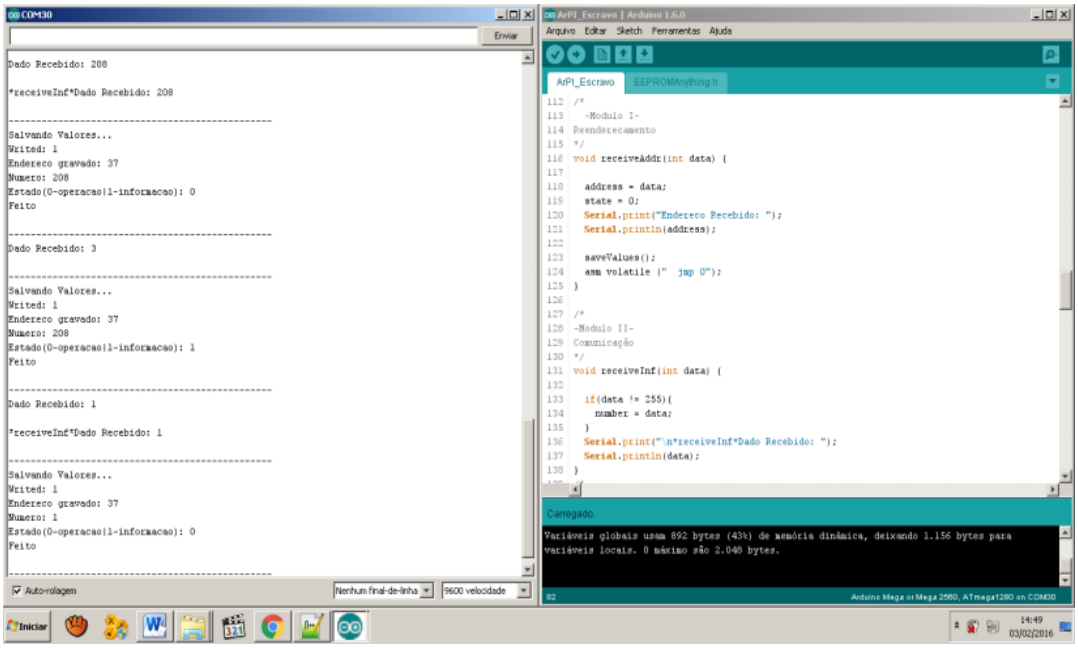

Figura 20 – Escravo recebendo um dado do mestre. Fonte: Elaborada pelo autor (2016).

4.2.4 Recuperar um dado armazenado

Conceitualmente uma comunicação é feita quando dois ou mais atores em um meio trocam informações. A troca é feita quando um dado/informação é enviado, processado e então devolvido ou recuperado. Esta operação é parte do processo de comunicação garantindo ao mestre (Raspberry PI) recuperar o dado de algum escravo (Arduino).

```
------------------------------------------------
Obtendo Valores...
Gravado(1- sim, 0- nao):1
Endereco guardado:21
Numero guardado:18
Estado guardado:0
Dados Recuperados!!
Feito!

------------------------------------------------
Ready!
```

Figura 21 – Dado armazenado no escravo (Arduino). Fonte: Elaborada pelo autor (2016).

Quando o usuário utiliza esta operação é requisitado ao mesmo que insira o endereço de algum dispositivo valido. Caso o dispositivo exista o sistema envia ao mesmo uma solicitação requisitando que ele envie ao mestre o dado que está armazenado em sua memoria interna. Esse dado deve ter sido previamente inserido dentro do Arduino, caso contrario ele retornará o valor 0 (zero).

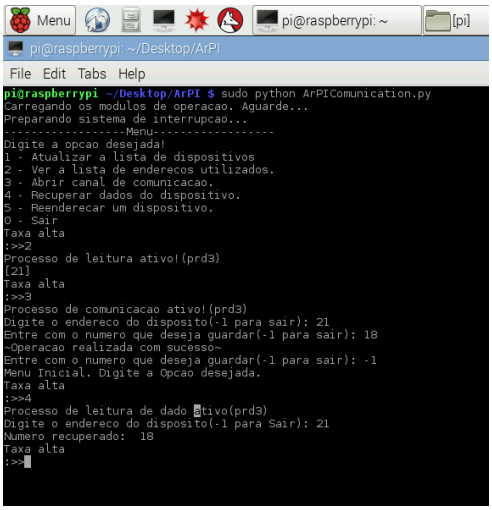

Figura 22 – Operação para recuperar um dado armazenado num escravo. Fonte: Elaborada pelo autor (2016).

4.2.5 Reendereçar

Esta operação permite ao usuário reendereçar um dispositivo atribuindo-lhe um endereço que não esteja em uso. Os endereços podem variar de 0 a 127, porém por convenção os endereços de 0 a 7 e de 120 a 127 foram reservados para uso interno do sistema. O restante é utilizado para endereçar os dispositivos em rede.

Quando esta operação é ativada o módulo de reendereçamento é executado perguntando ao usuário qual o dispositivo que ele quer reendereçar. O usuário entra com o endereço de um dispositivo valido e então o sistema procura-o na rede. Depois de localiza-lo o sistema pede ao usuário que insira o novo endereço. Se o endereço fornecido pelo usuário for valido e estiver disponível o sistema efetua as alterações.

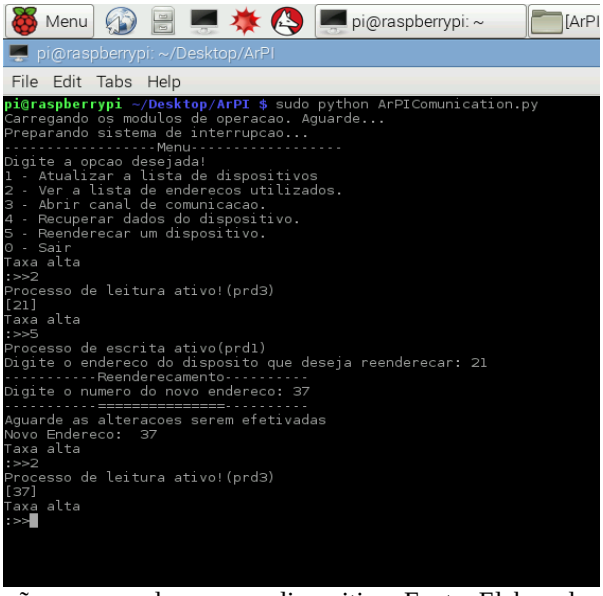

Figura 22 – Operação para reendereçar um dispositivo. Fonte: Elaborada pelo autor (2016).

Por meio do *serial motor* - mecanismo oferecido pelo *software* do Arduino- é possível observar as entradas e saídas na tela do computador que ele estiver conectado. Durante o processo de reendereçamento é fornecida uma saída através dessa ferramenta (*serial motor*) a fim de demonstrar os resultados do processo, conforme é mostrado na figura 23.

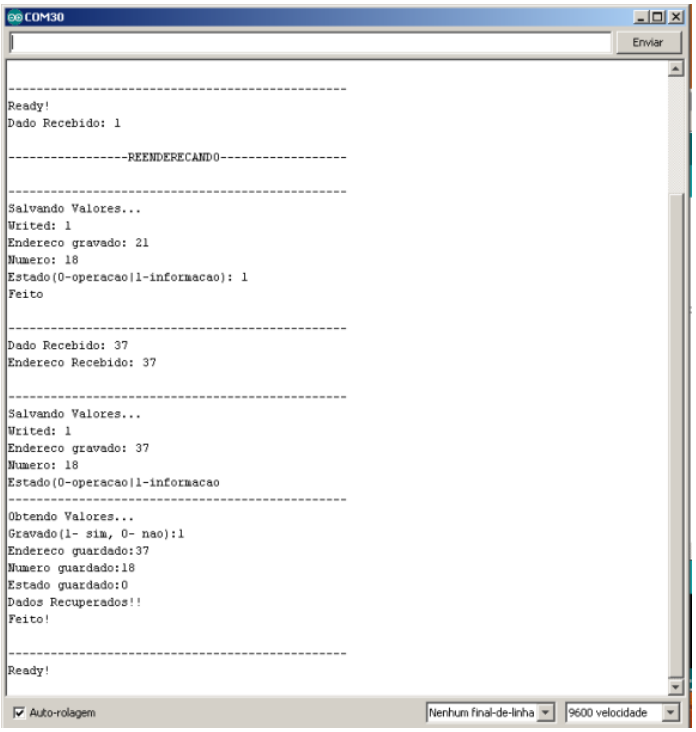

Figura 23 – Retorno do resultado do reendereçamento pelo Serial Motor do Arduino. Fonte: Elaborada pelo autor (2016).

4.3 Desafios da Interrupção

Dos módulos operacionais criados neste projeto, o mecanismo de interrupção foi o único que não pode ser implementado utilizando somente as linhas de comunicação conhecidas do I2C. Por este motivo a alternativa foi à utilização de um canal compartilhado que partia do mestre passando por todos os dispositivos escravos. Através desse canal seria permitido que qualquer dispositivo fizesse um pedido de interrupção ao mestre. O pedido de interrupção segue determinadas condições. Estas condições foram previamente estabelecidas com o proposito de padronizar e coordenar todo o processo.

O mecanismo de interrupção é funciona da seguinte forma:

- Foi inserido no mestre (*Raspberry* PI) um modulo que lê constantemente uma de suas portas programáveis. Esta porta permanece no *pull up*, isto é, o nível logico da porta é mantido alto. Observe na figura 24 a linha de interrupção em *pull up* representada pela linha vermelha, onde o nível logico do canal é mantido alto.

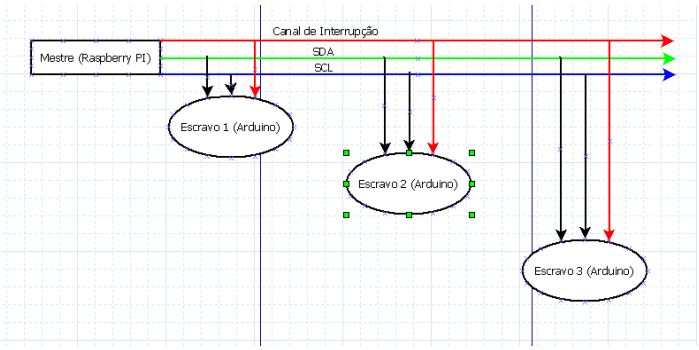

Figura 24 – Diagrama que mostra o canal de interrupção em *pull up.* Fonte: elaborada pelo autor (2016).

- Quando há uma mudança neste no nível logico deste canal um evento é detectado. Este evento é caracterizado como *pull down*, ou seja, quando o nível logico da porta cai de alto para baixo. Demonstrado pela figura 25, o *pull down* do canal de interrupção representado pela linha preta.

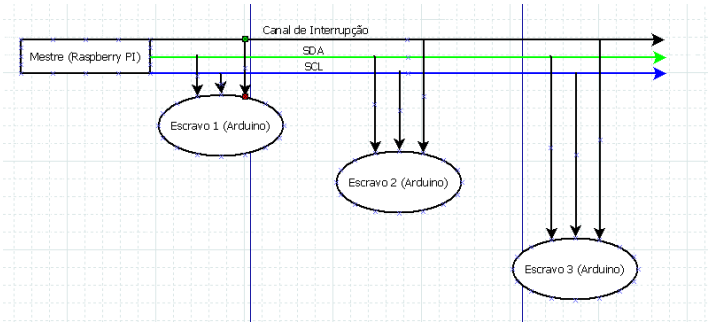

Figura 25 - Diagrama que mostra o canal de interrupção em *pull down.* Fonte: elaborada pelo autor (2016).

- Há um modulo no Arduino responsável por zerar o nível logico de uma de suas portas. Para fins de teste foi utilizado um botão para acessar este modulo. Quando este botão é pressionado o modulo zera o nível logico da porta e prepara o sistema do Arduino para enviar uma informação para sinalizar ao *Raspberry* PI que ele requisitou a interrupção.
- Assim que é detectado o *pull down* (transição da figura 24 para 25) o RPI para as suas atividades e salva o estado do sistema. Então é feito uma varredura na rede procurando qual dispositivo disparou o processo de interrupção.
- Depois de detectado o mestre (RPI) envia um dado ao requerente perguntando o que ele deseja.
- O escravo (Arduino) por sua vez envia ao mestre o pedido que deseja que ele execute.
- O processo é encerrado e o mestre volta ao estado anterior à interrupção

O principal desafio de inserir esse mecanismo no sistema foi à diferença de voltagem entre o *Raspberry* PI e os Arduinos, sendo obrigatório o uso de um conversor logico bidirecional de 3.3v-5v. Este conversor impede que os dispositivos se danifiquem.

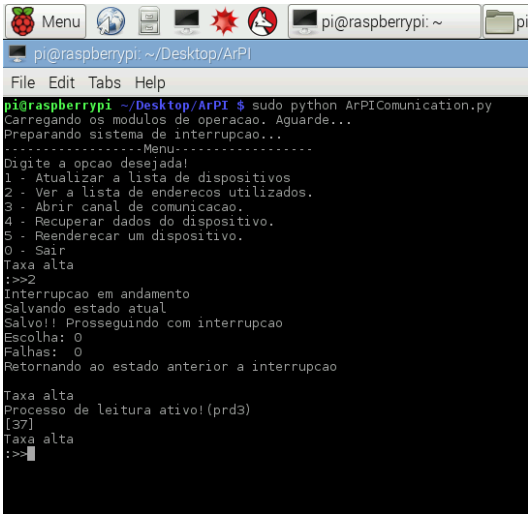

Figura 26 – Mecanismo de interrupção. Fonte: Elaborada pelo autor (2016).

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo a analise e a construção do protótipo do sistema de intercomunicação (ArPI) a fim de reunir as principais operações utilizadas em sistemas de comunicação. O protótipo do sistema ArPI reúne os principais módulos operacionais para estabelecer uma comunicação entre os dispositivos Arduinos e o *Raspberry* PI. A ideia é que estes módulos corroborem a hipótese da construção de uma rede mais ampliada e abrangente.

No decorrer do projeto foi possível observar que essas duas microcontroladoras trabalham muito bem juntas. O Arduino de um lado, sendo utilizado para automatizar as mais diversas atividades como, abrir e fechar portas, ligar e desligar aparelhos, acender e pagar luzes, entre muitas outras. E o *Raspberry* PI de outro, gerenciando todos esses processos e atividades. Com cada uma cumprindo seu papel, é possível conceber um sistema que gerencia operações de controle numa única plataforma.

Os resultados obtidos foram bastante promissores, demonstrado que é possível criar um sistema completo capaz de intercomunicar uma variedade significativa de dispositivos, fazendo-os trabalhar em conjunto. O protótipo cumpre muito bem com seu papel satisfazendo os objetivos principais estabelecidos no começo deste projeto.

Foi possível observar durante o desenvolvimento do protótipo do sistema ArPI a possibilidade de inserir novas opções e características. Ficando para trabalhos futuros a criação de uma interface gráfica interativa que possa disponibilizar para o usuário as opções do sistema de forma mais agradável e intuitiva. Também é possível apontar a possibilidade da criação de novas operações como: reiniciar um dispositivo remotamente, criar módulos operacionais nos escravos através do mestre e ativar e desativar estas operações. Também é sugerido como trabalhos futuros à implementação de fila de processos e controle de prioridades.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ADAFRUIT**, Products**. Disponível em < https://www.adafruit.com/products/1914> Acessado em 27 de novembro de 2015.

ADAM, **Using the I2C interface**. Disponível em <http://www.raspberry-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2> Acessado em 20 de dezembro de 2014.

ARDUINO, **What is Arduino?**. Disponível em: <https://www.arduino.cc/en/Guide/Introduction> Acessado em 08 de dezembro de 2014.

BYTEPARADIGM, **Introduction to I²C and SPI protocols**. Disponível em: <http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/>. Acessado em 10 de dezembro de 2014.

EDUARDO, L. **Reuso de requisitos para famílias de produtos em sistemas embarcados**, Revista da Universidade Ibirapuera, Jun 2011 n.1: 31-35.. Disponível em <http://www.ibirapuera.br/pesquisa/revista/vol1/capa.pdf#page=32> Acessado em 05 de dezembro de 2014.

GOTTFREDSON, L. **Mainstream Science On Intelligence**: An Editorial With 52 Signatories History, And Bibliography. Wall Street Journal, Dezembro 13, 1994. Disponível em <http://www.udel.edu/educ/gottfredson/reprints/1997mainstream.pdf> acessado em 08 de dezembro de 2014.

I2CBUS, **References**. Disponível em < http://www.i2c-bus.org/references/> Acessado em 19 de fevereiro de 2015.

LEE, E.A. **What's Ahead For Embedded Software**?. Ieee Computer, 2000;

LEMOS, M. **Conheça o Raspberry PI**. Disponivel em < http://blog.fazedores.com/conheca-o-raspberry-pi/> Acessado em 08 de janeiro de 2016.

MARWEDEL, P**. Embedded System Design**. Springer, 2003;

MAZIERO, C. A. **Sistemas Operacionais: Conceitos E Mecanismos**, 2013;

OLIVEIRA, A.C.O. **Tolerância A Falhas Para Sistemas Embarcados**, 2000;

PHILIPS SEMICONDUCTORS. **The I2C-bus specification**. Disponível em: <I2c2p.Twibright.Com/Spec/I2c.Pdf> Acessado em 05 de dezembro de 2014.

RASPBIAN, **Welcome to Raspbian,** 2016. Disponivel em <https://www.raspbian.org> Acessado em 27 de novembro de 2015

RASPBERRYPI, **About Us**. Disponível em < https://www.raspberrypi.org/about/> Acessado em 12 de outubro de 2015.

R.L.LATORRE. **Comunicação**. Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/Comunica%C3%A7%C3%A3o> acessado em 05 de dezembro de 2014.

SACCO, F. **Comunicação SPI**, 05 de maio de 2014. Disponível em: <http://www.embarcados.com.br/spi-parte-1/>. Acessado em 27 de novembro de 2015.

SILVA, F. **Introdução Aos Sistemas Distribuídos**, 2014.

SPENCER, S. **Private security.** ***On Patrol***, v. 1, n. 4, Winter 1996-97. Disponível em: <http://www.onpatrol.com/cs.privsec.html>. Acesso em: 28 nov. 1997.

TANENBAUM, A.S. **Sistemas Distribuídos**. 2ª Edição. Rio de Janeiro: Prentice-Hall, 2007.

WOLF, M. **Computers as Components: principles of embedded computing system design. Elsevier**: *United States of America*, 2012.

ZAMBARDA, F. **'Internet das Coisas': entenda o conceito e o que muda com a tecnologia.** Disponível em < http://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html> Acessado em 12 de janeiro de 2016.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Flávio dos S. Escobar,

autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação PROTOTIPO MODULAR DO SISTEMA DE INTERCOMUNICAÇÃO ARDUINO - RASPBERRY (ARPI)

de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 14 de OUTUBRO de 2016.

Flávio dos S. Escobar
Assinatura

Flávio dos S. Escobar
Assinatura