Departamento de Engenharia Elétrica
Rua Daniel Danelli s/nº Jardim Morumbi
Taubaté-Sp 12060-440
Tel.: (12) 3625-4190
E-mail: eng.eletrica@unitau.br

SEMÁFORO INTELIGENTE DE TRÂNSITO

Taubaté
2017

Paulo Gregório Soares Borba

Título: Semáforo Inteligente de Trânsito

Trabalho de Graduação apresentado ao Departamento de Engenharia Elétrica, da Universidade de Taubaté, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Marcio Abud Marcelino

Taubaté
2017

Ficha Catalográfica elaborada pelo SIBi – Sistema Integrado de Bibliotecas / UNITAU - Biblioteca das Engenharias

B726s

Borba, Paulo Gregório Soares
Semáforo inteligente de trânsito. / Paulo Gregório Soares Borba. – 2017.

59f. : il; 30 cm.

Monografia (Graduação em Engenharia Elétrica e Eletrônica) – Universidade de Taubaté. Departamento de Engenharia Elétrica e Eletrônica, 2017
Orientação: Prof. Dr. Marcio Abud Marcelino, Departamento de Engenharia Mecânica e Elétrica.

1. Semáforo inteligente. 2. Microcontrolador. 3. Tráfego. I. Título.

Aos meus pais, Paulo e Mariluce, por todo apoio e exemplo ao longo de minha vida.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, fonte da vida e da graça. Agradeço pela minha vida, minha inteligência.

Agradeço a minha família, meu pai Paulo, minha mãe Mariluce e minhas irmãs Ludmila e Carolina que me deram toda atenção, suporte, carinho para que eu pudesse concluir esta etapa da minha vida.

À todos os meus amigos, que estiveram presentes nesta etapa da minha vida e puderam compartilhar este sonho de se tornar engenheiro.

À Julia de Souza Pereira, por estar presente em todos os momentos, fáceis e difíceis, pelo apoio, cobrança e dedicação para que nada pudesse interferir no aprendizado e realização das tarefas.

Ao meu orientador, Prof. Dr. Marcio Abud Marcelino que jamais deixou de me incentivar. Sem a sua orientação, dedicação e auxílio, o estudo aqui apresentado seria mais difícil.

“A cultura forma sábios; a educação, homens.”

Louis Bonald

BORBA, P. G. S.; **Semáforo Inteligente de Trânsito.** 2017. 55 f. Trabalho de Graduação em Engenharia Elétrica – Departamento de Engenharia Elétrica e Eletrônica, Universidade Taubaté, Taubaté, 2017.

RESUMO

Este projeto apresenta um semáforo inteligente controlado por sensores de presença infravermelhos e um microcontrolador acoplado ao circuito para a execução da lógica e do controle, com custo reduzido. Atualmente os centros urbanos sofrem com o aumento significativo de veículos que circulam por suas vias, sendo que muitas destas não suportam o grande número de carros e motocicletas, ocorrendo assim engarrafamentos e tráfego lento, provocando acidentes com veículos e pedestres. Devido ao grande volume no trânsito, o emprego de semáforos comuns pode não ser viável, podendo, em algumas situações, até implicar em aumento no congestionamento. A solução que grandes centros urbanos estão encontrando para minimizar os efeitos do aumento do tráfego é a automatização de semáforos, chamados de semáforos inteligentes, que atuam segundo o volume de veículos encontrados nas ruas da cidade. Tal controle é efetuado por meio de sensores, câmeras de monitoração, radares e dados estatísticos de horários de picos. Para a execução do semáforo proposto foi utilizado o microcontrolador Arduino, e suas plataformas onde foram programadas as lógicas do projeto. Para a captação de presença dos veículos foi proposta a utilização o sensor infravermelho duplo DSE-830. A instalação de um sensor em cada via do cruzamento urbano, sendo que, na identificação de veículos por um dos sensores apenas a via do mesmo é priorizada, sendo que, na identificação dos veículos por ambos os sensores, o que em geral ocorre em horários de pico, o semáforo passa a operar com temporizações convencionais. O semáforo fica piscando em luz amarela durante tempo indeterminado se nenhuma das vias possuir tráfego e, a partir do momento em que um dos sensores detectar tráfego na via, o semáforo ligado a este sensor ficará verde enquanto o da outra via fechará, com a sinalização vermelha. Na captação do movimento o sensor utilizado irá detectar os veículos em um tempo de 3 segundos, sendo este tempo o necessário para a atuação de seu relé interno. O sensor possui dois canais com sensores piroelétricos, que se baseiam na detecção de calor emitido por objetos diversos, com ajustes independentes e ainda possui também um canal

sensor de micro-ondas podendo assim deixar o sensor com uma precisão maior para a aplicação no projeto. Pode-se concluir que este projeto apresenta uma melhor utilização da via, permitindo que a mesma não mantenha veículos parados, quando na outra via não trafegue nenhum automóvel e com sensores de custo reduzido o sistema proposto poderá ser empregado em qualquer rua e avenida urbana.

PALÁVRAS-CHAVE: Semáforo inteligente; Microcontrolador; Tráfego.

BORBA, P. G. S.; **Smart Traffic Light**. 2017. 55 f. Graduate work in Electrical Engineering - Department of Electrical and Electronic Engineering, University Taubaté, Taubaté, 2017.

ABSTRACT

This project presents an intelligent traffic light controlled by infrared presence sensors and a microcontroller coupled to the circuit for the execution of logic and control, that reducing the use of some traffic lights whose price is high. Currently, the urban centers suffer from a significant increase of vehicles that circulate along their tracks, where the most do not support the large number of cars and motorcycles, resulting in traffic jams and slow traffic, causing vehicular and pedestrian accidents. Due to the large volume of traffic, the use of common traffic lights may not be viable, and in some situations may even increase congestion. The solution that large urban centers are finding to minimize the effects of increased traffic is the automation of traffic lights, called intelligent traffic lights, which act according to the volume of vehicles found on the streets of the city. Such control is carried out by means of sensors, monitoring cameras, radars and statistical data of peak times, however the cost of the equipment used is high, and finishing that is often unviable for most cities. For the proposed semaphore, the Arduino microcontroller was used, and its platforms were used to program the project logic. The infrared sensor IRD-640 was used to capture the presence of the vehicles. The installation of a sensor in each road of the urban crossing, where, in the identification of one of the sensors only, the route of the same is prioritized, and in the identification of both sensors, which usually occurs at peak times, the traffic light starts to operate with conventional timings. The traffic light blinks in yellow light for an indeterminate time if none of the traffic routes are present, and once one of the sensors detects traffic on the road, the traffic light connected to this sensor will turn green while the other road will close, red. In motion capture the sensor used will detect the vehicles in a time of 3 seconds, this time being necessary for the operation of its internal relay. The sensor has two channels with pyroelectric sensors, which are based on the detection of heat emitted by various objects, with independent settings and also has a microwave sensor channel, thus allowing the sensor with a greater accuracy for the application in the project. It can be concluded that this project presents a better use of the road, allowing it not to keep vehicles stopped, when in the

other road does not traffic any car and still feasible the cost of the equipment, so that it can be used in any street and avenue urban.

Keywords: Smart traffic light; Microcontroller; Traffic.

LISTA DE FIGURAS

Figura 1 – Arduino UNO 20
Figura 2 – Esquematização eletrônica completa da plataforma Arduino 22
Figura 3 – Estrutura interna do microcontrolador Atmega 328 23
Figura 4 – Pinagem do microcontrolador ATMega328 24
Figura 5 – Exemplo de declarações dos pinos da plataforma utilizando void setup 29
Figura 6 – Exemplo de aplicação de constantes e variáveis em um programa de arduino 31
Figura 7 – Exemplificação do uso de comandos de funções de controle de fluxo 33
Figura 8 – Exemplo de aplicação *IF/ELSE IF* e *SWITCH/BREAK* 34
Figura 9 – Exemplo de aplicação de comandos *While* e *Do..While* 35
Figura 10 – Espectro Eletromagnético 37
Figura 11 – Ilustração de um sensor infravermelho 39
Figura 12 – Sensores piroelétricos de detecção infravermelha 39
Figura 13 – Esquema elétrico dos sensores piroelétricos HC-SR501 40
Figura 14 – Esquema básico de funcionamento de um sensor microondas 41
Figura 15 – Sensor infravermelho e micro-ondas DSE 830 42
Figura 16 – Esquema de ligação do circuito do projeto 44
Figura 17 – Interface de programação IDE – *Software* Arduino 46
Figura 18 – Interface do *software* UnoArduSim utilizada para simular o programa 48
Figura 19 – Fluxograma completo 50

LISTA DE TABELAS

Tabela 1 – Descrição dos pinos do microcontrolador ATMega PD0 a PD7 25
Tabela 2 – Descrição dos pinos do microcontrolador ATMega PB0 a PB7..... 26
Tabela 3 – Descrição dos pinos do microcontrolador ATMega PC0 a PC6 26
Tabela 4 – Funções matemáticas e de tempo no Arduino 32
Tabela 5 – Especificações técnicas do sensor DSE 830 43
Tabela 6 – Tabela verdade do funcionamento do circuito..... 51

LISTA DE ABREVIATURAS E SIGLAS

ANSI *American National Standards Institute* – Instituto Nacional Americano de Padrões
CPU *Central Processing Unit* – Unidade de processamento central
DC *Direct Current* – Corrente Direta
DSL *Domain Specific Language* – Linguagem de domínio específico
EPROM *Erasable programmable ready-only memory* – Memória programável apagável somente leitura
IDE *IntegratedmDevelopment Environment* – Ambiente de desenvolvimento integrado
LED *Ligth Emitting Diode* – Diodo emissor de luz
PWM *Pulse Width Molulation* – Modulação por largura de pulso
SRAM *Static Random Access Memory* – Memória estática de acesso aleatório

SUMÁRIO

1 INTRODUÇÃO 16
1.1 OBJETIVO DA PESQUISA 17
1.2 DELIMITAÇÃO DA PESQUISA 17
1.3 JUSTIFICATIVA DA PESQUISA 17
1.4 ESTRUTURA DO TRABALHO 17
2 REVISÃO BIBLIOGRÁFICA 19
2.1 MICROCONTROLADOR 19
2.2 MICROCONTROLADOR X MICROPROCESSADOR 19
2.3 MICROCONTROLADOR ARDUÍNO UNO 19
2.3.1 COMPOSIÇÃO DO ARDUINO UNO 20
2.3.2 MICROCONTROLADOR ATMEGA328 23
2.3.3 ALIMENTAÇÃO 26
2.3.4 MEMÓRIA 27
2.3.5 ENTRADAS E SAÍDAS 27
2.3.6 PROGRAMAÇÃO ARDUINO UNO 28
2.3.6.1 INSTRUÇÕES DE PROGRAMAÇÃO NO ARDUINO 28
2.4 OSCILADOR DE CRISTAL 35
2.5 SENSORES 36
2.6 SENSOR INFRAVERMELHO 37
2.7 SENSORES MICROONDAS 41
3 PROCEDIMENTO METODOLÓGICO 44
3.1 MATERIAIS 44
3.2 FUNCIONAMENTO 49
4 RESULTADOS OBTIDOS 52
5 CONCLUSÃO 53
ANEXO 1 – PROGRAMAÇÃO BASEADA EM C++ DO PROJETO NO ARDUINO 54
REFERÊNCIAS 59

1 INTRODUÇÃO

Atualmente o controle de tráfego viário está se tornando ultrapassado e inviável devido ao aumento de veículos nos centros urbanos, gerando assim o aumento do trânsito e tornando as vias públicas caóticas e perigosas para motoristas e pedestres.

Este projeto deu continuidade ao Trabalho de Conclusão de Curso de Arantes (2010) que usou câmeras de vídeo para detecção de presença, e automatizar o emprego de semáforos convencionais de trânsito, melhorando o tráfego e reduzindo o consumo de combustível dos veículos, que permanecem parados nos semáforos desnecessariamente. O estudo se baseia na plataforma microcontroladora ARDUINO UNO, que utiliza o microcontrolador ATMEGA328, que tem função de ser o dispositivo *master* para o funcionamento do sistema.

Este trabalho usou sensor infravermelho para realizar a função das câmeras de monitoração, porém com o custo mais baixo, e permitindo que o mesmo seja instalado fora do alcance de vândalos, e evitando possíveis burlas do sistema. Este trabalho também analisou o trabalho de conclusão de curso de Mattos e Manteli (2006) que também estudou o emprego de sensores óticos para a captação de veículos nas vias urbanas, porém que foram instalados ao alcance de pessoas por não possuírem longo alcance de atuação. Na presença de pedestres que venham a atravessar a via onde os sensores infravermelhos serão, não deve responsabilizá-los, por motivos de segurança, fazendo com que o pedestre tenha que apertar um botão externo para que o semáforo possa fechar para que possa circular com segurança na via pública.

Neste trabalho, a sensibilidade de presença do veículo foi realizada por sensores tipo infravermelho e micro-ondas, que é responsável por captar a presença de veículos nas vias, não importando a quantidade deles. Enquanto uma das vias permanecer vazia a outra estará com o semáforo aberto e assim se aplica para a outra, e quando as duas possuírem veículos o sistema controla os semáforos de forma convencional, temporizando-os. O sistema não permite a captação de pequenos objetos, como pessoas ou animais que venham a circular na via pública e, caso algum dos sensores entrar em modo de falha o sistema controla os semáforos de forma convencional temporizada, evitando assim a má atuação, ou alguma falha decorrente do funcionamento equivocado dos sensores, garantindo uma falha segura.

1.1 OBJETIVO DA PESQUISA

O objetivo principal deste trabalho é proporcionar um tráfego com maior qualidade, mais fluente, reduzindo acidentes envolvendo grandes quantidades de veículos e pedestres, e encurtando o tempo de viagem de veículos que venham circular pela mesma.

1.2 DELIMITAÇÃO DA PESQUISA

Este trabalho foi apresentado como um estudo que visa principalmente melhorar a qualidade do trânsito em centros urbanos, adaptando um semáforo convencional de trânsito, temporizado, em um semáforo inteligente, de forma que enquanto nas vias não existir tráfego, os semáforos ficarão piscando as lâmpadas amarelas até que em uma das vias se detectar algum veículo. Então nesta via o semáforo ficará verde e o segundo fechará, ficando vermelho, e assim simultaneamente para ambas as vias. Quando existir tráfego constante nas duas vias os semáforos funcionarão de forma convencional, temporizada.

1.3 JUSTIFICATIVA DA PESQUISA

A motivação para este projeto é aplicar uma nova forma de detecção de presença de veículos, utilizando meios mais rentáveis e de baixo custo, possibilitando assim o emprego em qualquer via pública e melhorar a qualidade do trânsito das cidades.

1.4 ESTRUTURA DO TRABALHO

Este projeto apresenta a seguinte disposição de capítulos:

A apresentação da plataforma microcontroladora utilizada, o Arduino UNO, o seu respectivo microcontrolador ATMEGA328, definição de programação em linguagem C e a definição de sensores infravermelhos e de micro-ondas estão no capítulo 2.

Nos capítulos 3 é apresentada breve introdução sobre a aplicação do sistema, os materiais que foram utilizados e a programação utilizada no microcontrolador ATMEGA328 presente na plataforma Arduino UNO.

No capítulo 4 são apresentados os resultados encontrados.

Finalmente, no capítulo 5 são apresentadas as conclusões, juntamente com as sugestões em futuros trabalhos que possam melhorar o sistema.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentado um referencial teórico sobre o que é, e o conceito de funcionamento de circuitos integrados microcontroladores. Será apresentada a plataforma de programação Arduino, sensores infravermelhos e micro-ondas.

2.1 MICROCONTROLADOR

Microcontrolador é um circuito integrado (CI), considerado um computador e possui a finalidade de controlar dados através de uma porta. De acordo com Ferreira (2015), esses CIs são considerados “pastilhas inteligentes, dotadas de um processador, pinos de entrada/saída e uma memória”.

Já para Neto, Monteiro e Queiroga (2012) é definido como microcontrolador um pequeno componente eletrônico capaz de controlar processos lógicos a partir de uma inteligência programável. Possui internamente memória de dados, portas de entrada, *timers*, saída paralela, entre outros.

2.2 MICROCONTROLADOR X MICROPROCESSADOR

Um microcontrolador pode ser diferenciado de um microprocessador levando-se em conta muitos aspectos, sendo que a principal diferença é nas suas funcionalidades. Para que se possa usar um microprocessador, se faz necessário a adição de outros componentes para se operar em conjunto com o mesmo, tais como memória e outros componentes que são usados para receber e enviar dados para o mesmo. No aspecto geral, o microprocessador é o coração de um computador.

Já para o microcontrolador, que é um dispositivo que foi projetado inicialmente para executar a função de todos os componentes necessários para que um microprocessador funcione, ou seja, é um componente mais completo. Desta maneira, o microcontrolador se torna mais barato e mais eficaz, tanto por sua funcionalidade, quanto ao seu modo de aplicação (ASSIS, 2004).

2.3 MICROCONTROLADOR ARDUÍNO UNO

A plataforma microcontroladora Arduino UNO, é uma pequena interface que possui em sua placa o microcontrolador ATMega328, e foi criado em 2005 no *Ivrea Interaction Design Institute*, com intuito de facilitar a interação desta nova tecnologia

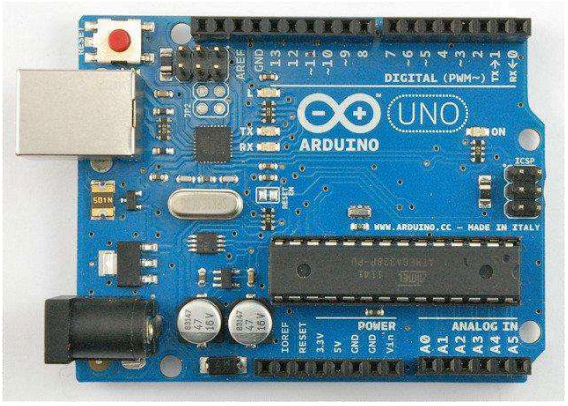
na época à estudantes sem experiência em eletrônica e programação (MCROBERTS, 2011).

O sistema microcontrolador Arduino UNO vem sendo utilizado como recurso para muitas aplicações desde a sua criação e foi adotado para a aplicação deste projeto controlando via sensores infravermelhos semáforos de trânsito.

Para Mélo et al. (2011) "*a plataforma Arduino é "uma plataforma baseada em software e hardware de fácil utilização, de prototipação de fonte aberta e pode ser utilizada por artistas, técnicos e qualquer pessoa que venha a se interessar em criar ambientes interativos e objetos diversos utilizando a tecnologia simples do equipamento*."

Sobre as plataformas Arduino, por se tratarem de *hardware* e *software* livres, é possível encontrar diversos tipos de plataformas para diferentes tipos de aplicação, cada uma delas com uma vantagem, desenvolvidas com uma eletrônica simples para que possa ser modelada caso haja necessidade. Existem várias plataformas Arduino, distintas, com aplicações específicas, envolvendo uso de tecidos, e até com *bluetooth*. A Figura 1 apresenta o microcontrolador Arduino Uno que foi utilizado para a aplicação do projeto deste artigo.

Figura 1: Arduino UNO



Fonte: Arduino (2015).

2.3.1 COMPOSIÇÃO DO ARDUINO UNO

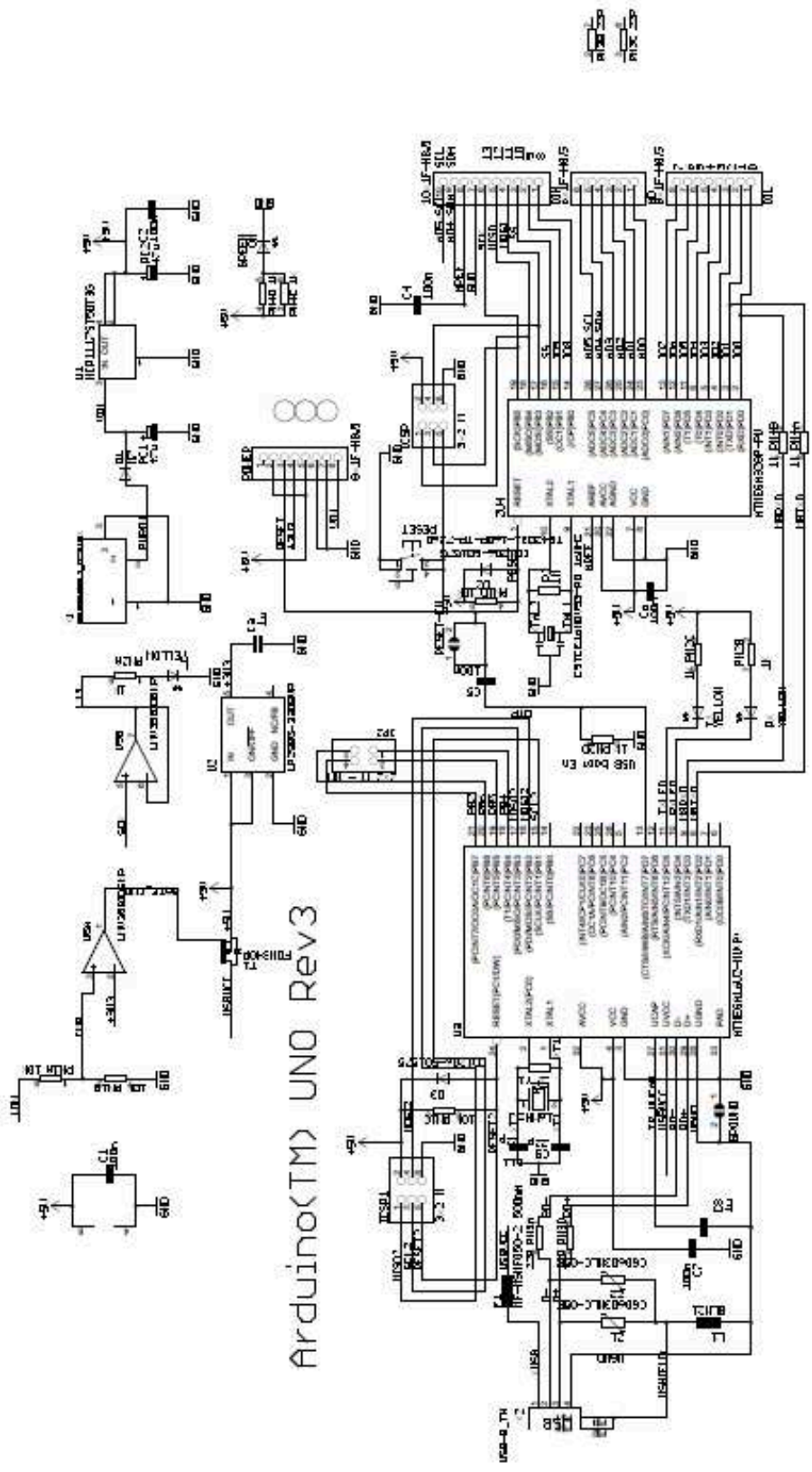
A plataforma Arduino possui diversos seguimentos, variando dependendo da aplicação. Para melhor familiarização com a placa, está descrito a seguir a

funcionalidade de cada característica da mesma, sendo composta de (FILHO, 2012). A Figura 2 apresenta o circuito da plataforma Arduino.

Características da plataforma Arduino UNO:

- Microcontrolador - ATMega328
- Tensão de Operação – 5 Volts
- Tensão nas *Inputs* (recomendado) – 7 a 12 Volts
- Limites nas *Inputs* – 6 a 20 Volts
- Pinos de Entradas/Saídas Digitais – 14 (6 saídas PWM)
- Pinos de Entradas Analógicas – 6
- Corrente DC (*Direct Current)* por pino de Entrada/Saída – 40 mA (mili Ampére)
- Memória *Flash* – 32KB referente ao microprocessador ATMega328
- SRAM (*Static Random Access Memory*) – 2KB
- EPROM (*Erasable programmable ready-only memory*) – 1KB
- *Clock Speed* – 16 MHz

Figura 2: Esquematização eletrônica completa da plataforma Arduino



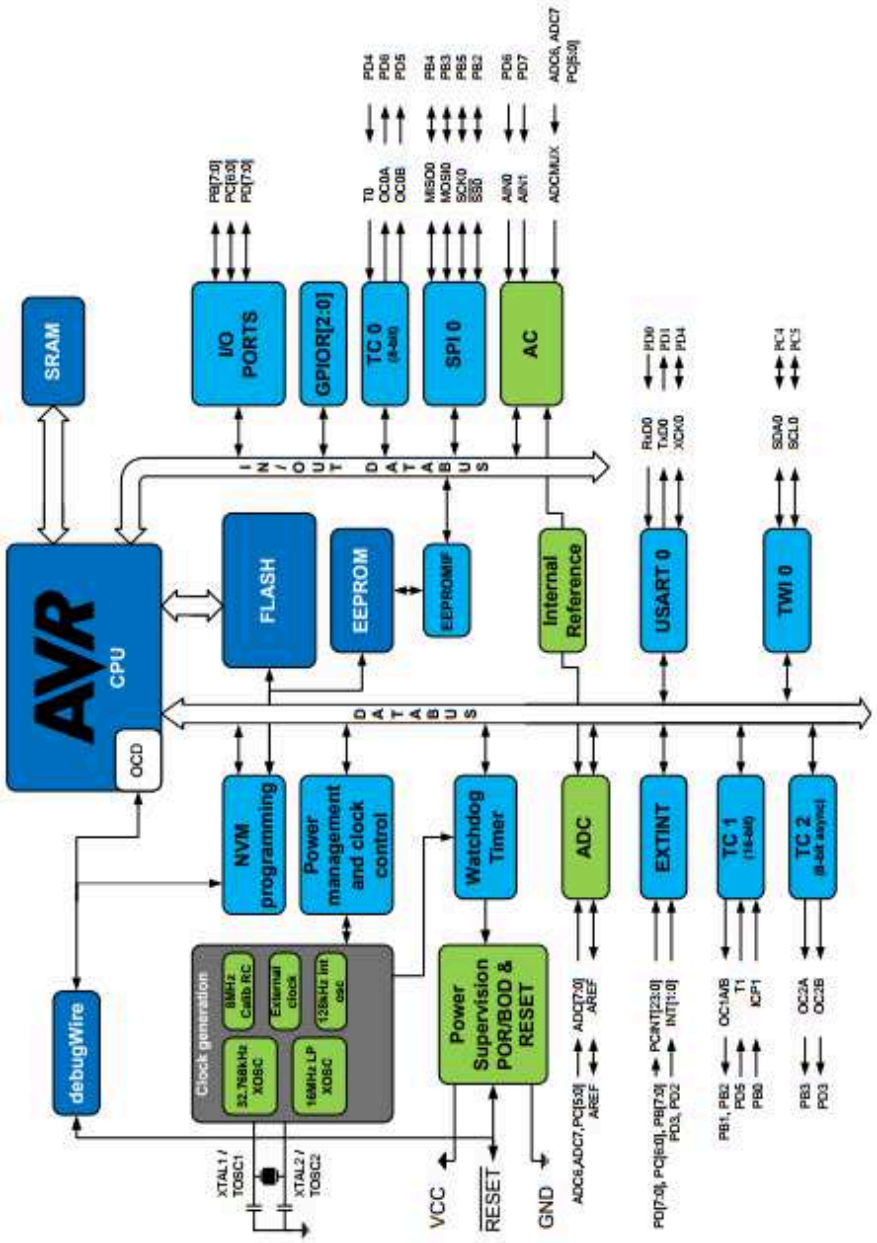
Fonte: Filho (2012).

2.3.2 MICROCONTROLADOR ATMEGA328

O microcontrolador ATMega328 é um componente da família de CI's (*Integrated Circuit*) de 28 pinos, e foi criado pelo grupo ATMEL e é utilizado nas plataformas Arduino. Este pequeno *chip* é um microprocessador de 8 bits, que dispõe de arquitetura Harvard modificada, que é o modo que o microcontrolador acessa os dados em sua memória interna.

O modelo da arquitetura de um microcontrolador ATMega328 está ilustrado na Figura 3.

Figura 3: Estrutura interna do microcontrolador Atmega 328:

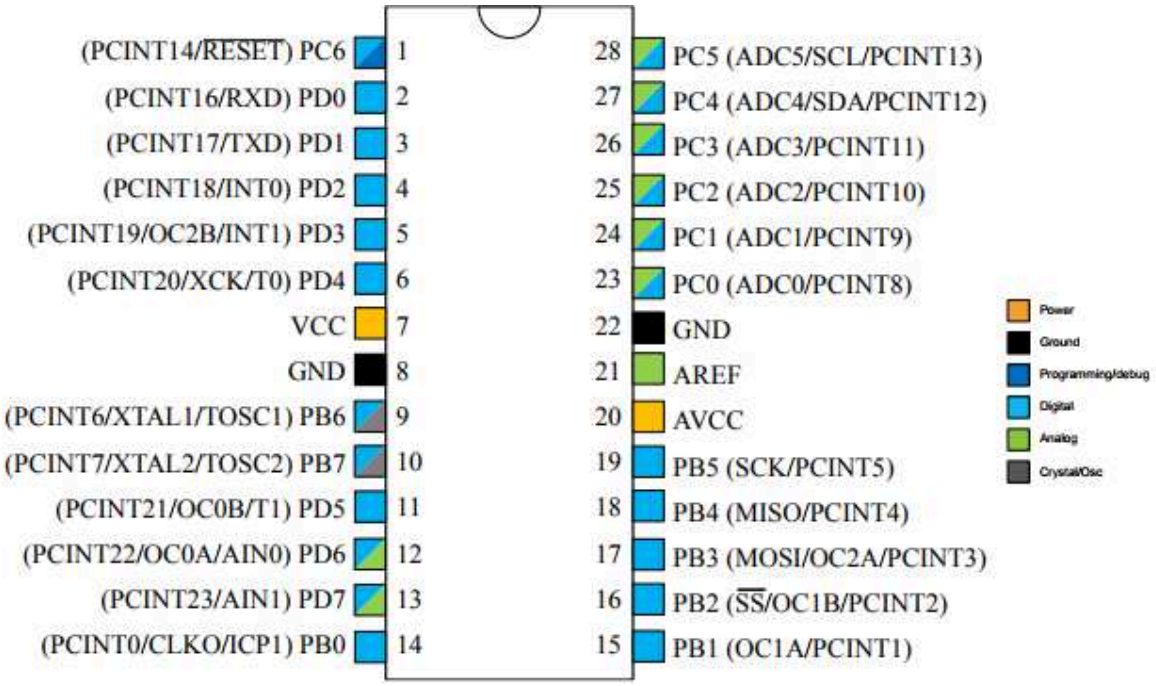


Fonte: ATMEL (2016).

Todo microprocessador necessita de uma área interna dedicada à sua CPU (*Central Processing Unit*), a qual é responsável por executar a maior parte dos cálculos internos que o mesmo venha a ser submetido (CAETANO, 2007).

A seguir, é descrita a função de cada pino do microcontrolador ATMega328 na Figura 4.

Figura 4: Pinagem do microcontrolador ATMega328



Fonte: ATMEL (2016).

De acordo com a figura 15 pode-se introduzir a explicação completa do esquema de pinos do microprocessador. O componente dispõe de 28 pinos os quais são:

PC6 – *Reset*: é responsável por efetuar o *reset* do estado em que o microprocessador se encontra, seja durante a execução de uma tarefa ou quando ocorrer alguma falha. Normalmente neste canal é ligado a um sinal positivo através de um resistor e no outro contato do resistor é ligado a saída do botão que é responsável por gerar o pulso de *reset*. No outro terminal do botão é ligado a um sinal negativo, portanto quando atuado o botão ele faz a conexão do terminal 1 com o negativo, assim efetuando o *reset* do microcontrolador (BERTOGNA, 2013).

PD0-7: são portas de 8 bits direcionais de entrada e saída, sendo que cada porta possui uma função alternativa, que pode-se observar na Tabela 1.

Tabela 1: Descrição dos pinos do microcontrolador ATMega PD0 a PD7

Pinos	Funções Alternativas	
PD0	RXD	Entrada de Recepção Serial
PD1	TXD	Saída de Transmissão Serial
PD2	INT0	Entrada Interrupção Externa INT0
PD3	INT1	Entrada Interrupção Externa INT1
	OC2B	Saída de Comparação B do T/C2
PD4	T0	Entrada Contagem do T/C0
	XCK	Entrada/Saída Clock Externo da USART
PD5	T1	Entrada Contagem do T/C1
	OC0B	Saída de Comparação B do T/C0
PD6	AIN0	Entrada Positiva Comparador Analógico
	OC0A	Saída de Comparação A do T/C0
PD7	AIN1	Entrada Negativa Comparador Analógico

Fonte: BERTOGNA (2013).

VCC (Voltagem Corrente Contínua) – este canal é dedicado para a alimentação do circuito integrado. Sua tensão de alimentação pode variar de 1,8 a 5,5 V em corrente direta.

GND (*ground)* – este canal é dedicado à referência (pino terra).

PB0-7 – como na configuração dos pinos de PD0-7 da tabela 1, estes canais também são portas de 8 bits bidirecionais de entrada e saída, e possuem uma função alternativa, como pode ser representada na Tabela 2.

PC0-6: diferente das portas PD e PB, estas portas possuem 7 bits bidirecionais de entrada e saída, sendo que o pino PC6 já foi citado. O restante dos canais podem ser descritos na Tabela 3.

Tabela 2: Descrição dos pinos do microcontrolador ATMega PB0 a PB7

Pinos	Funções Alternativas	
PB0	CLK0	Saída de clock dividido
	ICP1	Entrada de Captura do T/C1
PB1	OC1A	Saída Comparação A do T/C1
PB2	OC1B	Saída Comparação B do T/C1
	$\overline{SS}$	Seleção Master/Slave da SPI
PB3	OC2A	Entr. Inv. do Comp. Analógico
	MOSI	Saída Output/Slave da SPI
PB4	MISO	Saída Input/Slave da SPI
PB5	SCK	Entrada de Clock da SPI
PB6	XTAL1	Entrada Clock Externo 1
	TOSC1	Entrada Clock T/C2 qdo Osc. RC
PB7	XTAL2	Entrada Clock Externo 2
	TOSC2	Entrada Clock T/C2 qdo Osc. RC

Fonte: BERTOGNA (2013).

Tabela 3: Descrição dos pinos do microcontrolador ATMega PC0 a PC6

Pinos	Funções Alternativas	
PC0...PC5	ADC0...ADC5	Entradas conversores A/D
PC4	SDA	Entrada/Saída de Dados da I2C
PC5	SCL	Entrada/Saída de Clock da I2C
PC6	$\overline{RESET}$	Entrada de Reset

Fonte: BERTOGNA (2013).

AVCC: este canal é dedicado à tensão de entrada do conversor Digital/Analógico, e deve ser conectada externamente a entrada comum VCC.

AREF: este pino tem a característica da referência para a entrada de conversor A/D (Analógico/Digital) (BERTOGNA, 2013).

2.3.3 ALIMENTAÇÃO

A alimentação da plataforma Arduino Uno pode-se fazer de duas maneiras. A primeira delas é utilizando a porta USB presente no *hardware* da placa, ou também

utilizando uma fonte de alimentação externa, que pode ser subdividida em duas maneiras, ou através de baterias ou um adaptador AC-DC.

Segundo as características do *hardware* apresentadas no capítulo 2.4, quando se utiliza uma alimentação externa, a plataforma permite operar-se com uma alimentação entre 6 a 20 volts, porém é recomendado que se utilize uma alimentação de 7 a 12 volts para manter uma condição não agressiva para o circuito da plataforma (HOEPERS, 2012).

2.3.4 MEMÓRIA

Segundo Arduino (2011), o microcontrolador ATMega328 utilizado pela plataforma Arduino Uno possui uma memória de 32 KB, sendo que 0,5 KB dessa memória é utilizada para o *bootloader*, que é um gerenciador de boot instalado no microcontrolador e é responsável por gerenciar os programas que serão executados em determinados ciclos de tempo. O microcontrolador também possui 2 KB de SRAM e 1KB de memória EEPROM, que podem ser manipuladas utilizando a biblioteca.

Caso a memória disponibilizada pelo microcontrolador não seja suficiente para a aplicação, é possível utilizar um banco de memória adicional com uma plataforma *Shield* que seja capaz de se adicionar um MicroSD (*Solid Disc)* (HOEPERS, 2012).

2.3.5 ENTRADAS E SAÍDAS

O modelo da plataforma Arduino Uno possui em seu *hardware* 14 pinos digitais, que podem ser configurados como entradas digitais ou saídas digitais por meio da programação inserida no microcontrolador. Também possui 6 entradas analógicas que são nomeadas de A0 a A5, onde podem ser ligados os sinais analógicos que serão digitalizados e lidos pelo microcontrolador utilizando instruções de programação, para se dimensionar a grandeza do valor que pode ser inserido através destes pinos.

Ainda com relação aos pinos que a plataforma possui, pode-se destacar o pino 13, que é um pino que possui uma função específica, ou seja, a função de um LED, (diodo emissor de luz).

De todos os pinos digitais e analógicos, pode-se utilizar um total de 6 pinos para se realizar a função de PWM (*pulse width modulation*). A função de PWM consiste numa técnica para ter resultado analógico com significado digital, variando-se o sinal

ligado-desligado para se gerar uma onda quadrada que pode simular valores de 0 a 5 volts através da mudança do tempo de atuação entre os estados de ligado e desligado (HOEPERS, 2012).

2.3.6 PROGRAMAÇÃO ARDUINO UNO

A plataforma Arduino é uma plataforma de *hardware* livre, ou seja, pode ser modificada por qualquer pessoa. A programação é feita em um *software* com base IDE (*Integrated Development Environment*). Neste *software* é escrita toda a programação necessária para o funcionamento correto da plataforma.

O circuito da plataforma Arduino é baseado em entradas e saídas simples, que serão controladas e desenvolvidas a partir de uma biblioteca que utiliza uma linguagem baseada em C++. A linguagem de programação do Arduino está relacionada a linguagem *Wiring* e o ambiente para desenvolvimento baseia-se no *Processing* (PINTO, 2014).

Durante a criação de uma programação utilizando este microcontrolador, existe a necessidade do contato entre o computador e a plataforma Arduino através do IDE, utilizando uma *Domain Specific Language* (DSL), que usa a linguagem baseada em C++ como dito anteriormente. Nesta plataforma é possível verificar pinos de entradas com conversores analógico-digital, portas digitais, saídas analógicas, comunicação serial (ARAÚJO, 2012).

2.3.6.1 INSTRUÇÕES DE PROGRAMAÇÃO NO ARDUINO

Neste capítulo, é apresentado um pouco mais sobre a linguagem de programação C++ que se utiliza para estruturar um programa na interface IDE, para o funcionamento do microcontrolador Arduino.

Segundo Casavella (2017) a linguagem C foi desenvolvida em 1972, nos laboratórios da companhia *Bell Telephone*, por Dennis Ritchie, com o propósito de ser utilizada para desenvolver uma versão nova do sistema *Unix*, pois a primeira versão utilizava-se de linguagem *Assembly*. Este tipo de linguagem foi criada por programadores para programadores.

Esta linguagem foi padronizada pela ANSI (*American National Standards Institute*), para que se pudesse ser utilizada de forma padrão por todos que viessem

a programar utilizando-a em microcontroladores, *drivers* e outros controladores de dispositivo, na criação de sistemas operacionais, etc.

Algumas das funções do programa Arduino podem parecer difíceis ao primeiro contato, porém quando se começa a utilizá-lo, pode-se perceber que é bem simples de se programar. São estes os principais comandos e funções que se utiliza para programar o microprocessador:

- Void Setup () – Esta função é a primeira a ser chamada no cabeçalho do programa, e é a primeira função que o microprocessador irá executar. Esta função tem por objetivo informar qual a configuração e comportamento de seus pinos através do comando *pinMode* ao longo da programação, e também é responsável por inicializar a porta serial da plataforma (SILVEIRA, 2012).

- Void Loop() – Este comando se dá logo após o *Setup()*. Quando se programa algo nesta função, todas serão repetidas a cada ciclo do programa. Esta função permanece lendo os pinos de entrada e comandando as saídas da placa e sua porta serial (SILVEIRA, 2012).

A Figura 5 apresenta um exemplo de programação básica utilizando estas duas funções para definir as funções dos pinos da plataforma:

Figura 5: Exemplo de declarações dos pinos da plataforma utilizando void setup.

```
void setup(){
  pinMode(sem1vermelho, OUTPUT);          // Define característica do pino
  pinMode(sem1amarelo, OUTPUT);           // Define característica do pino
  pinMode(sem1verde, OUTPUT);             // Define característica do pino
  pinMode(sem2vermelho, OUTPUT);          // Define característica do pino
  pinMode(sem2amarelo, OUTPUT);           // Define característica do pino
  pinMode(sem2verde, OUTPUT);             // Define característica do pino
  pinMode(sensor1, INPUT);                // Define característica do pino
  pinMode(sensor2, INPUT);                // Define característica do pino
}
```

Fonte: Autor (2017).

Na programação do Arduino, existem dois tipos de dados que podem ser trabalhados, variáveis e constantes:

- Variáveis – são alocadas em posições da memória de programa e são marcadas por um nome que pode ser definido pelo programador e o tipo de dado que estas irão armazenar. Durante o ciclo do programa estas memórias podem receber outros valores. Podem receber valores iniciais ou permanecer vazias. Existem quatro tipos de formato de variáveis que podem ser declaradas no início da programação: byte, int, long e float. Na declaração de uma variável, quando a mesma não necessita de um retorno de valor, utiliza-se o prefixo de programação void (SILVEIRA, 2012).

- Constantes – este tipo de dado, é aquele que uma vez definido não pode ser alterado nunca. São representados por três grupos, *True/False*, *High/Low* e *Input/Output* que podem ser representadas pelos valores em formato numérico binário 0 ou 1 (SILVEIRA, 2012).

A Figura 6 apresenta um exemplo de aplicação dos dados constante e variável. Neste exemplo é possível observar o uso de um comando muito utilizado dentro das programações de microcontroladores, que é o comando *delay*. Este comando é utilizado para se aplicar um tempo dentro da execução do programa. Sempre que um comando deste é utilizado, junto a ele é inserido um valor numérico de unidade de grandeza de mili segundos, ou seja, quando executado dentro de um programa, o valor que acompanha a função é a quantidade de tempo que o programa espera para executar sua próxima função.

Figura 6: Exemplo de aplicação de constantes e variáveis em um programa de Arduino

```
/* Esse programa escrito em C do Arduino aumenta e diminui gradativamente o brilho de um LED
conectado no pino PWM 10 do Arduino. */

int i=0;                                  // declaração da variável global inteira i iniciada com 0
void ledOn( );                            // declaração da função criada ledOn do tipo void
void setup( )  {
        pinMode(10,OUTPUT); // aqui 2 parâmetros são passados à função pinMode( )
}
void loop( )    {
        for (i=0; i <= 255; i++) ledOn( );          // aumenta o brilho do led
        for (i=255; i >= 0; i--) ledOn( );          // diminui o brilho do led
}
void ledOn( ) {                                     // função que acende o led
        analogWrite (10, i);      // o nº do pino e o valor de i são passados à função analogWrite( )
        delay (10);
}
```

Fonte: Silveira (2012).

- Matrizes – estas funções são usadas para armazenar várias variáveis do mesmo tipo, portanto são posições dentro da memória de programa, onde existem endereços que podem ser acessados utilizando um identificador, que pode-se chamar de Índice (SILVEIRA, 2012).

- Operações Aritméticas e Lógicas – pode-se realizar operações aritméticas e lógicas dentro de uma programação, sendo adição, subtração, divisão e multiplicação representadas respectivamente pelos símbolos +, -, / e *, sendo que os operandos devem estar separados. Em uma programação dentro de uma plataforma é muito comum ser utilizados os recursos de comparadores de valores ou variáveis utilizando os símbolos =, < e >. Podem ser realizados também cálculos lógicos a partir de portas lógicas sendo &&, || e ! para portas E (*and*), OU (*or*) e NÃO (*not*) respectivamente (SILVEIRA, 2012).

    - Funções Matemáticas e de Tempo – são utilizadas diversos comandos matemáticos e de tempo dentro de uma programação na plataforma Arduino. A Tabela 4 apresenta estas funções (SILVEIRA, 2012).

Tabela 4: Funções matemáticas e de tempo no Arduino

Função	Exemplo	Notas
delay(ms) Essa função pausa o programa por um período em milissegundos indicado pelo parâmetro entre parênteses.	**delay(1000);** Com esse parâmetro o programa vai pausar durante 1 segundo (1000 ms).	Durante o período em que essa função está ativa qualquer outra função no programa é suspensa; é equivalente ao HALT em Assembly. Somente as interrupções de hardware podem parar essa função.
delayMicroseconds(us) Essa função pausa o programa por um período em microssegundos indicado pelo parâmetro entre parênteses.	**delayMicroseconds(1000);** Com esse parâmetro o programa vai pausar durante 1 ms (1000 us).	As mesmas observações acima para a função **delay(ms)** são válidas aqui.
millis( ) Retorna o número de milissegundos desde que o Arduino começou a executar o programa corrente.	**long total = millis( );** Aqui a variável inteira longa (de 32 bits) 'total' vai guardar o tempo em ms desde que o Arduino foi inicializado.	Essa variável vai ser resetada depois de aproximadamente 9 horas.
random(min,max) Gera números pseudo-aleatórios entre os limites **min** e **max** especificados como parâmetros.	**int valor = random(100,400);** À variável 'valor' vai ser atribuido um número inteiro qualquer entre 100 e 400.	O parâmetro **min** é opcional e se excluído o limite mínimo é 0. No exemplo variável 'valor' poderá ser qualquer número inteiro entre 0 e 400.
abs(x) Retorna o módulo ou valor absoluto do número real passado como parâmetro.	**float valor = abs(-3.14);** Á variável 'valor' vai ser atribuído o número em ponto flutuante (e sem sinal) 3.14.	
map(valor,min1,max1,min2,max2) A função **map( )** converte uma faixa de valores para outra faixa. O primeiro parâmetro 'valor' é a variável que será convertida; o segundo e o terceiro parâmetros são os valores mínimo e máximo dessa variável; o quarto e o quinto são os novos valores mínimo e máximo da variavel 'valor'.	**int valor = map(analogRead(A0),0,1023,0,255));** A variável 'valor' vai guardar a leitura do nível analógico no pino A0 convertida da faixa de 0-1023 para a faixa 0-255.	Com essa função é possível reverter uma faixa de valores, exemplo: **int valor = map(x,1,100,100,1);**

Fonte: Silveira (2012)

- Funções para controle de fluxo – são utilizadas para selecionar uma ou mais instruções baseando-se no processo de comparação e seu resultado. Funciona como uma pergunta dentro do processo que será executado, e é representada pelo comando IF (se) e o comando ELSE (senão). Esta função não necessariamente precisa estar representada pelas duas palavras de comando, porém deve ser sempre representada pela palavra IF, que dá o início da comparação (SILVEIRA, 2012). A Figura 7 apresenta um exemplo da utilização

de uma função de controle de fluxo com somente uma palavra e também com duas palavras de programação:

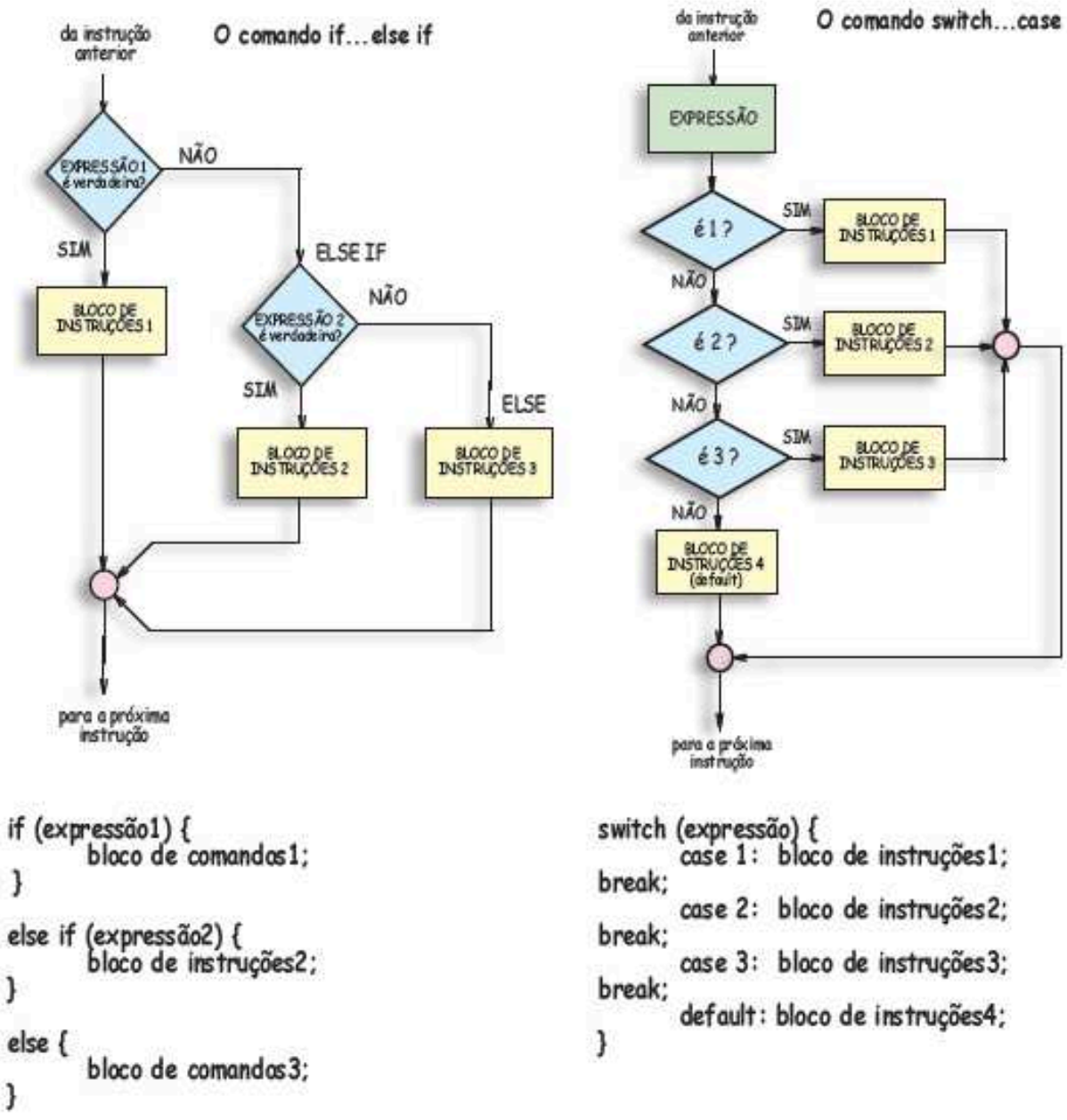
Figura 7: Exemplificação do uso de comandos de funções de controle de fluxo

```
include <stdio.h>
main()
{
int a,b;
printf("Digite 2 números: ");
scanf("%d %d",&a,&b);
if (b)
  printf("%f",a/b);
else
  printf("Nao posso dividir por zero\n");
}
```

Fonte: Trevisan (2014)

É possível também se criar outro comando IF dentro de outro comando IF, assim cria-se uma cadeia de comparações, sem que se precise separar todas elas como no exemplo da Figura 8 utilizando os comandos *IF/ELSE IF* ou o comando *SWITCH/BREAK* (SILVEIRA, 2012).

Figura 8: Exemplo de aplicação *IF/ELSE IF* e *SWITCH/BREAK*



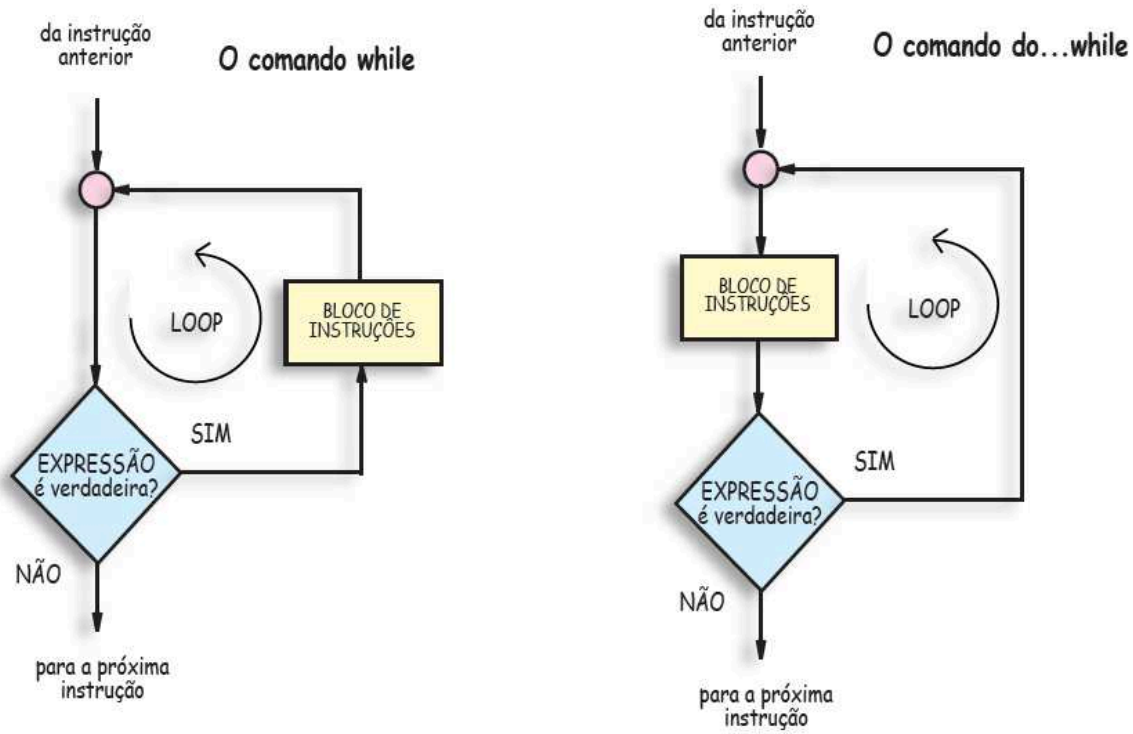

```
if (expressão1) {
        bloco de comandos1;
}
else if (expressão2) {
        bloco de instruções2;
}
else {
        bloco de comandos3;
}
```

```
switch (expressão) {
        case 1:  bloco de instruções1;
break;
        case 2:  bloco de instruções2;
break;
        case 3:  bloco de instruções3;
break;
        default: bloco de instruções4;
}
```

Fonte: Silveira (2012).

Outra função de controle de fluxo que pode ser programada na linguagem C é o comando While, que é uma das operações mais frequentes que os programas dos microprocessadores executam. É utilizada para repetir um grupo de instruções até que uma condição que inicialmente foi verdadeira se torne falsa, e também pode-se utilizar de um complemento que é o comando *Do...While*, onde o bloco de instrução é deslocado ao início da caixa de condição para que a instrução seja executada ao menos uma vez (SILVEIRA, 2012), conforme ilustrado na figura 9.

Figura 9: Exemplo de aplicação de comandos *While* e *Do..While*:



Fonte: Silveira (2012).

2.4 OSCILADOR DE CRISTAL

Oscilador de cristal é um bloco eletrônico utilizado para sincronizar com precisão o tempo (*Clock*) das operações de um microcontrolador.

Os osciladores de cristal não são somente utilizados em microcontroladores. Estes pequenos componentes estão presentes em muitos equipamentos como relógios, computadores, cronômetros, equipamentos de comunicação, entre outros (BRAGA, 2017).

Os cristais, em geral, de Quartzo, são materiais piezoelétricos que apresentam uma polarização elétrica em suas faces, se sofrer deformações. Além de possuírem esta propriedade, estes componentes são altamente ressonantes, ou seja, quando sofrem alguma deformação mecânica, por exemplo, estes materiais produzem uma vibração em uma frequência natural. Portanto, quando aplicada uma tensão entre os terminais de um cristal de quartzo, o mesmo se deforma de acordo com a intensidade deste potencial, e esta deformação faz com que o material produza uma vibração que

é muito precisa. Então, a partir desta vibração pode-se fazer cálculos de tempo e outros tipos de operação (BRAGA, 2017).

2.5 SENSORES

Algo muito importante para a realização deste trabalho, são os chamados sensores, responsáveis por detecção de presença de veículos.

Os dispositivos sensores são componentes quem possuem a capacidade de mudar o seu estado quando submetidos a variação de alguma grandeza física, tornando-o capaz de perceber alterações no meio externo e, desta maneira, converter tal variação em sinais elétricos que podem ser mensurados. Tais componentes vão dos mais simples modelos, como chaves mecânicas de atuação, até sensores mais sofisticados como por exemplo sensores que são capazes de detectar a porcentagem de oxigênio em algum meio externo (HOEPERS, 2012).

Dentro do mundo da automação, os sensores podem ser classificados tomando em conta suas características. São eles: sensores de contato, de proximidade e distância, sensores de uso diversos e sensores de visão.

A seguir uma breve explicação sobre cada um destes tipos de sensores:

- Sensores de contato – este tipo de sensor é capaz de alterar seu estado de acordo com alguma atuação física dele para com o meio, ou seja, se algo tocar o sensor aplicando uma força ele é capaz de fechar um contato fazendo com que este sinal possa prosseguir e assim acionar alguma válvula, etc. Na maioria dos casos são sensores eletromecânicos pois necessitam de um contato físico para que ele possa funcionar corretamente.

- Sensores de proximidade e distância – são sensores capazes de detectar um objeto próximo a ele, não havendo assim, contato mecânico entre o meio e o sensor. Alguns modelos são capazes de medir a distância entre o objeto detectado e o sensor. Estes tipos de sensores são denominados sensores de alcance.

- Sensores de tipos diversos – esta categoria de sensores, incluem alguns tipos de sensores especiais, utilizado para medir pressão, temperatura, fluxo de fluidos, gases, etc.

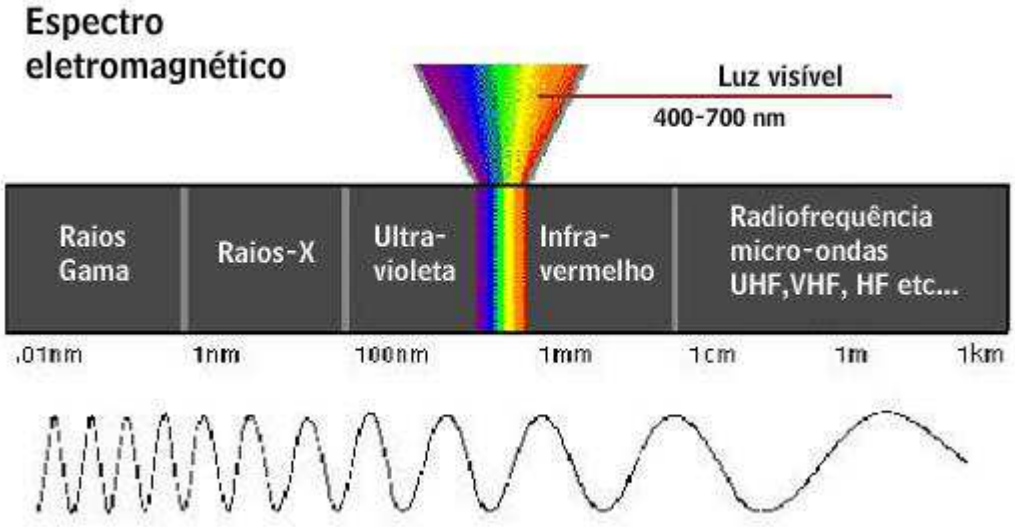
- Sensores de visão – são capazes de captar imagens de processos e a partir delas extrair informações que auxiliam na orientação do processo. Este tipo de sensor geralmente é usado na indústria da robótica, automobilística, etc, a fim de serem utilizados em tarefas de inspeção, reconhecimento de materiais, entre outras.

A partir desta variedade de componentes disponíveis no mercado, é possível realizar um projeto, tendo em vista a facilidade de emprego das aplicações com a utilização destes sensores (HOEPERS, 2012).

2.6 SENSOR INFRAVERMELHO

É um tipo de sensor que detecta e/ou emite a faixa de radiação de luz infravermelha que está presente no espectro eletromagnético que é formado por todos os comprimentos de ondas possíveis (PALHETA, 2012), conforme ilustrado na Figura 10.

Figura 10: Espectro Eletromagnético



Fonte: PALHETA (2012).

Os sensores infravermelhos são classificados de duas maneiras, passivos e ativos, sendo que os sensores passivos somente detectam a radiação infravermelha e o sensor infravermelho ativo possui a capacidade de emitir a onda de radiação infravermelha (MAZZAROPPI, 2007). Quando se fala em sensor infravermelho ativo, pode-se subdividir este tipo de sensor em outros três tipos que vão de acordo com a aplicação. São eles: sensores por sistema de barreira, por sistema de difusão e por reflexão.

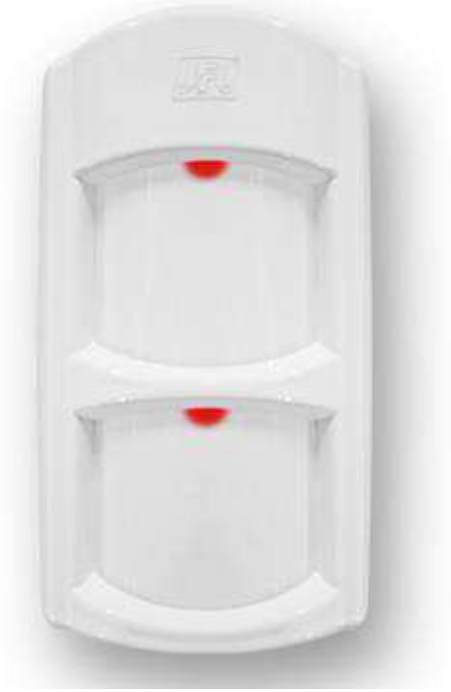
- Sistemas de barreira – este sensor emite a faixa da radiação e é posicionado exatamente em frente ao sistema receptor, em uma distância pré determinada de acordo com cada tipo de sensor, e será detectado quando algo invadir esta barreira entre o emissor e o receptor.

- Sistemas por difusão – nestes sensores, os raios que são emitidos pelo emissor são direcionados para o objeto a ser detectado e assim é gerado um feixe de luz infravermelha que chega ao receptor, que conseguirá calcular a distância de acordo com o tempo em que o sinal volta para o mesmo, entre a emissão do feixe e a recepção.

- Sistemas por reflexão – este sistema utiliza ao invés de um receptor, um espelho prismático, que atua quando houver uma interrupção por algum objeto entre o feixe e o espelho (HOEPERS, 2012).

A Figura 11 apresenta o sensor JFL, modelo IRD 640. Este sensor possui dois sensores do tipo difusão piroelétricos para maior segurança na detecção de presença. É um sensor microcontrolado que analisa os sinais dos sensores, e trata informações como amplitude do pulso, polaridade do pulso, e intervalo de repetição para validar o resultado e impedir que o sensor envie disparos falsos (JFL, 2007)

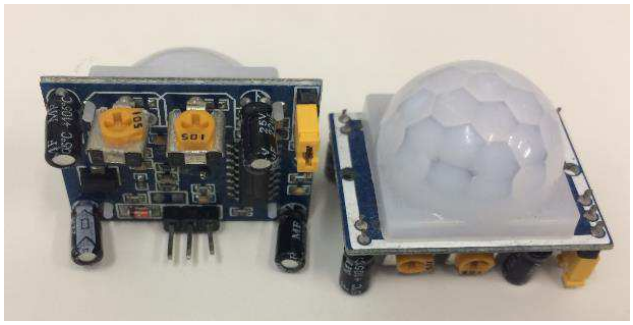
Figura 11: Ilustração de um sensor infravermelho



Fonte: JFL (2007)

A Figura 12 apresenta dois sensores piroelétricos detectores de movimento por difusão do tipo HC-SR501, do fabricante *ShenzenChipskey Technology, Limited*, e foi o sensor utilizado para a simulação deste projeto.

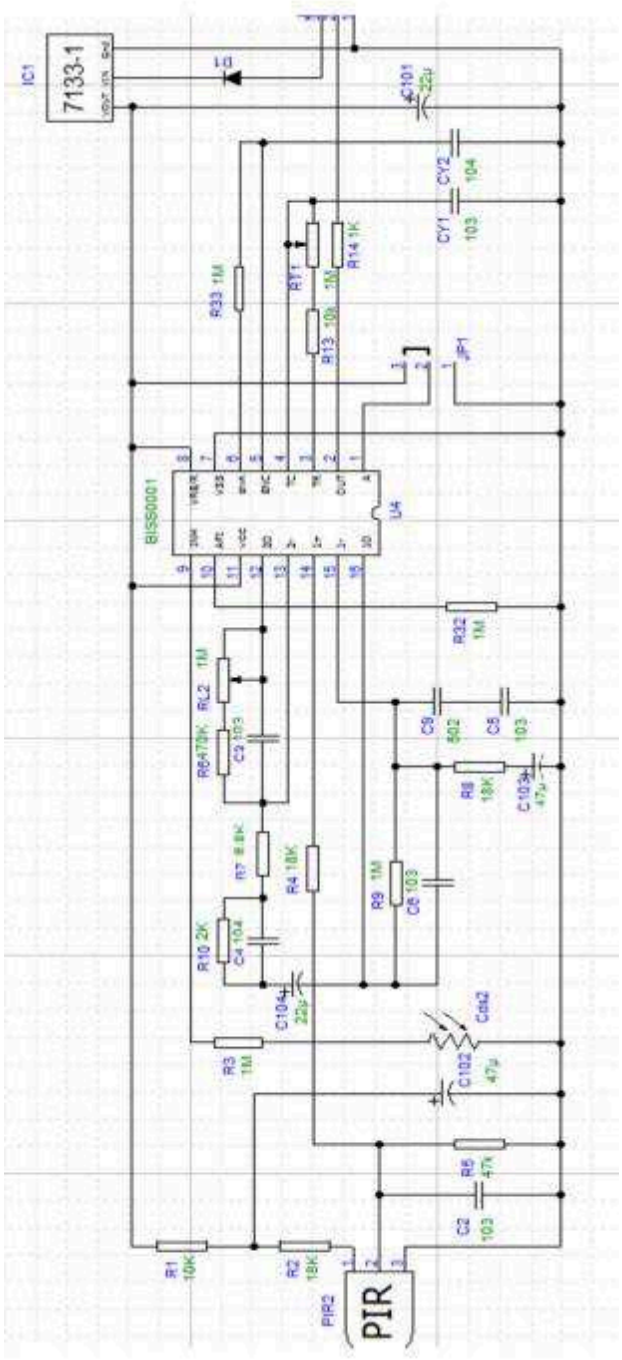
Figura 12: Sensores piroelétricos de detecção infravermelha



Fonte: SC Technology (2011).

A Figura 13 apresenta o circuito eletrônico presente na placa do sensor piroelétrico da figura 25.

Figura 13: Esquema elétrico dos sensores piroelétricos HC-SR501



Fonte: HC-SR501 *Datasheet* (2011).

Em geral, todos os objetos emitem alguma forma de radiação térmica, na maioria das vezes dentro do espectro infravermelho, porém esta radiação é invisível para os olhos, mas pode ser detectada por este tipo de sensor. Este sensor capta a radiação de um objeto que venha a atingir seu campo de atuação, assim o mesmo capta tal radiação por meio de componentes piroelétricos, que são materiais que possuem a

capacidade de gerar uma tensão elétrica quando o algo invade o campo emitido pelo sensor causando uma variação da distância da luz infravermelha que retorna para o sensor de maneira que o circuito do mesmo dispare e ligue um sinal para o meio externo (MAZZAROPI, 2007).

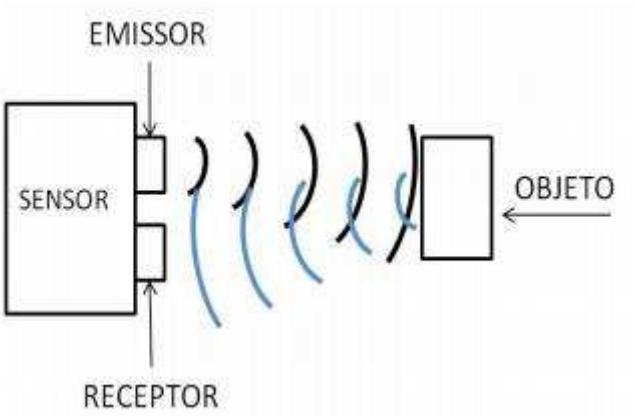
2.7 SENSORES MICROONDAS

Os sensores microondas sãos sensores muito utilizados na área de segurança por possuírem alta capacidade de detecção e sensibilidade. O funcionamento desse sensor é basicamente semelhante ao funcionamento de um sensor infravermelho ativo, porém, a radiação por ele emitida está na faixa de microondas no espectro de radiações eletromagnéticas (OLIVEIRA, et al. 2013).

A radiação eletromagnética emitida pelo sensor é refletida pelo objeto, e captada de volta através de uma antena que está presente no sensor, assim, desta maneira, essa radiação que é captada de volta é misturada com a radiação eletromagnética original por meio de diodos Schottky, fazendo com que o sinal de saída do sensor atue. (OLIVEIRA, et al. 2013)

A Figura 14 apresenta o esquema de funcionamento de um sensor micro-ondas.

Figura 14: Esquema básico de funcionamento de um sensor microondas



Fonte: OLIVEIRA, et al. (2013).

O sensor DSE 830 da marca JFL apresentado na figura 15, proposto para a aplicação do projeto é um sensor que apresenta tanto funcionamento utilizando ondas eletromagnéticas da faixa do infravermelho quanto da faixa de microondas. Possui 2 sensores piroelétricos, e um terceiro sensor micro-ondas, fazendo com que a precisão

de captação de objetos seja mais precisa e segura, evitando uma possível falha na detecção.

Este sensor é recomendado para este projeto, embora tenha custo superior em aos sensores IRD 640 e HC-SR501 ilustrados nas figuras 11 e 12, pois oferece maior confiabilidade por possuir sensibilidade tripla e poder ser instalado em áreas externas. Este sensor possui a capacidade de ser ajustado para áreas externas e pode ser regulado a fim de evitar que seja disparado por objetos ou reações indesejadas, como pessoas, chuvas e ventos.

Figura 15: Sensor infravermelho e micro-ondas DSE 830

Fonte: JFL (2017).

As especificações técnicas do sensor DSE 830 podem ser expressada através da tabela 5 retirada do *datasheet* do componente disponibilizado no site do fabricante.

Tabela 5: Especificações técnicas do sensor DSE 830

DSE-830	
Sensor Piroelétrico	Dois canais com sensor piro digital com duplo elemento
Led de indicação	5
Microondas	1 canal 10.525 Ghz
Imunidade PET	Até 30 Kg
Níveis de sensibilidade do Piroelétrico	3 nível de sensibilidade
Níveis de sensibilidade do Microondas	Sim
Chave Tamper	Sim
Ângulo de detecção	ângulo 90º
Alcance de detecção	Até 14 metros
Compatibilidade	Centrais convencionais e monitoradas
Função Bootloader	Sim
Características Gerais	
Condições de operação	0°C ~ +50°C (32°F ~ 122°F)
Tensão de operação	9Vdc ~ 18 Vdc
Consumo em Stand by	Max. 56,7 mA (51,4 mA em Stand by)
Dimensão	70 x 125 x 55 mm
Peso	96g

Fonte: JFL (2017).

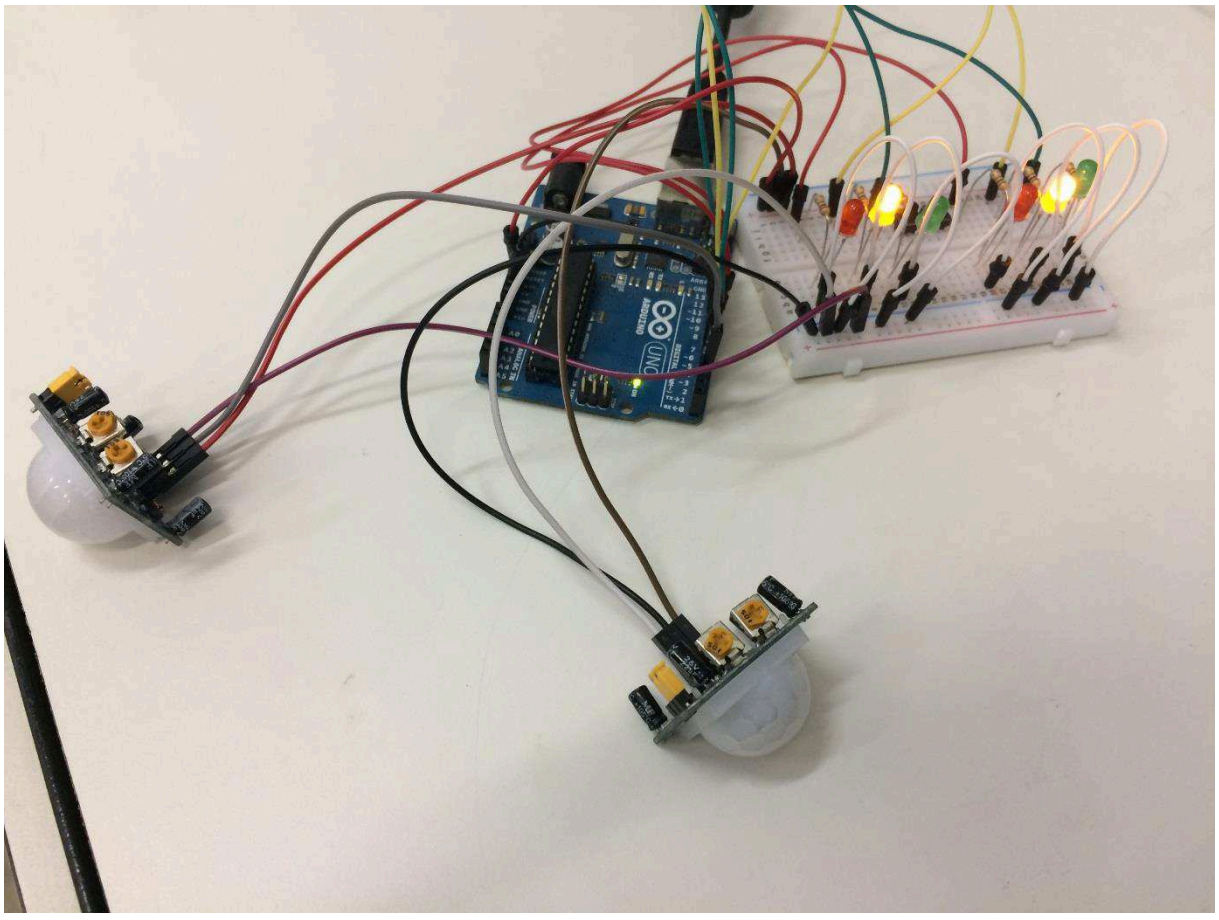
3 PROCEDIMENTO METODOLÓGICO

3.1 MATERIAIS

A seguir, segue a descrição dos materiais utilizados para a realização de um protótipo que serviu para demonstrar a operação do sistema.

No sistema denominado semáforo inteligente de trânsito existe um conjunto de componentes que atuam no funcionamento do mesmo. São eles: o microcontrolador Arduino, uma placa *protoboard* onde é realizada a conexão dos componentes sensores e os LED's que representam os semáforos. A figura 16 apresenta o circuito do projeto.

Figura 16: Esquema de ligação do circuito do projeto



Fonte: Autor

Os sensores escolhidos para esse protótipo foram apresentados na figura 12, e têm a função de captar a presença dos veículos nas vias urbanas e, a partir desse

sinal o microcontrolador executa a programação e controla o funcionamento das lâmpadas de ambos os semáforos do cruzamento simples.

Para controlar o funcionamento das lâmpadas dos semáforos (LED's), foram utilizados os pinos de saídas digitais da plataforma Arduino UNO, resistores de 620 Ω(OHMS) ilustrados na figura 16.

A plataforma Arduino foi conectada ao computador via USB, com a finalidade de se transferir a programação em linguagem C++ para a placa.

O *software* de programação em C++ possui uma interface ilustrada na figura 17.

Figura 17: Interface de programação IDE – *Software* Arduino

Fonte: Autor

O *software* foi obtido através do site do desenvolvedor oficial da plataforma Arduino (www.arduino.cc). Após a escolha da plataforma a ser utilizada no projeto, foi iniciado o processo de programação da lógica a ser carregada no microcontrolador através de uma comunicação USB entre o computador e a placa.

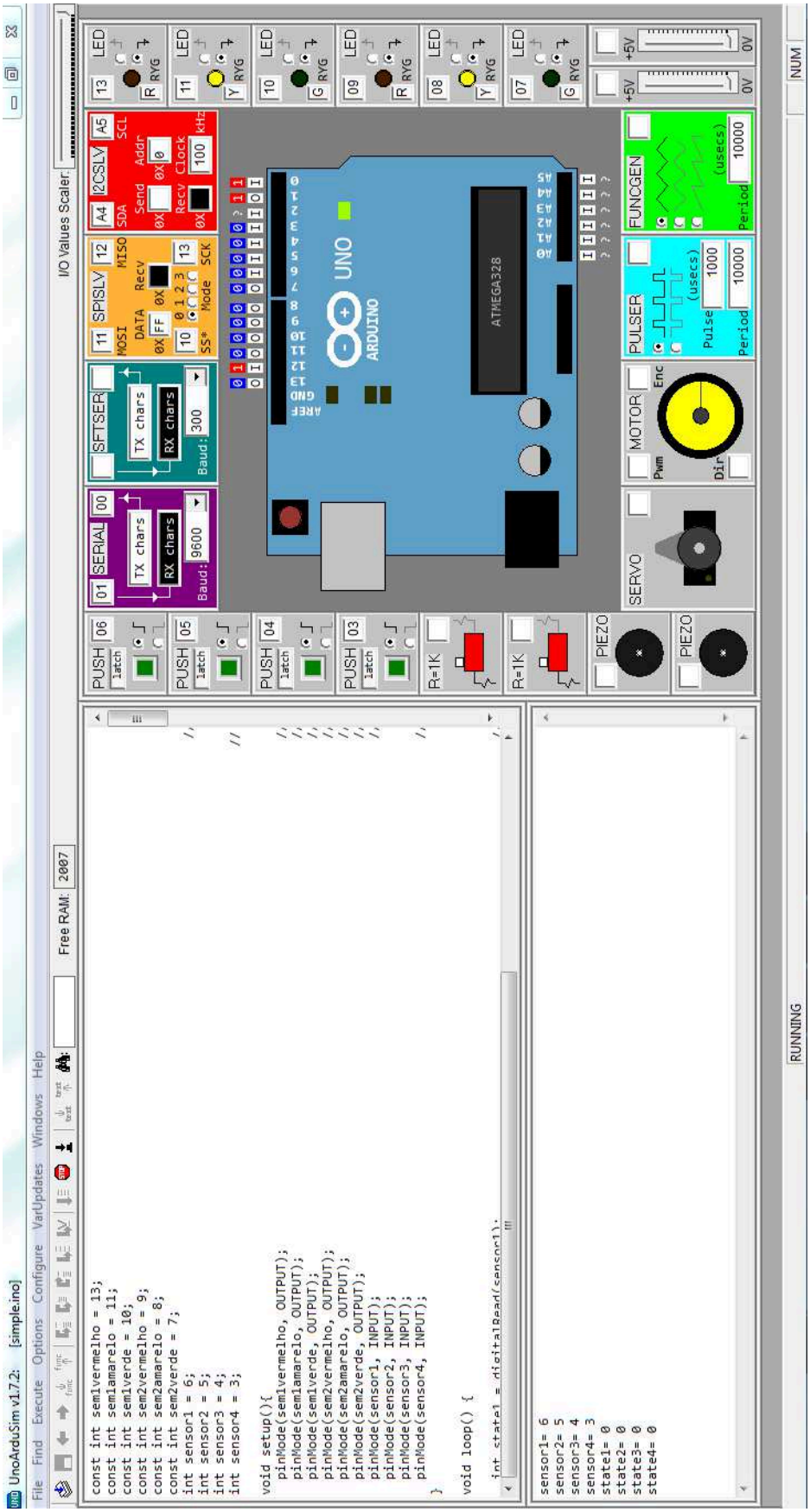
Dentro da programação do Arduino, é padrão ser utilizado uma estrutura onde se declara todas as variáveis e constantes e depois suas funções. São elas as funções setup e loop.

Dentro da função setup foi indicado quais as características dos pinos da placa a serem utilizados, e dentro da função loop foram escritas as funções, as instruções a sequência do programa.

O código programado para que o objetivo do projeto fosse cumprido pôde ser simulado através de um *software* chamado UnoArduSim, obtido na internet de modo gratuito. Com ele foi possível testar as diversas condições que o circuito pode ser submetido e também foi possível otimizar e corrigir falhas dentro da programação.

A Figura 18 apresenta a interface do *software* utilizada para simular a aplicação do projeto.

Figura 18: Interface do *software* UnoArduSim utilizada para simular o programa



Fonte: Autor

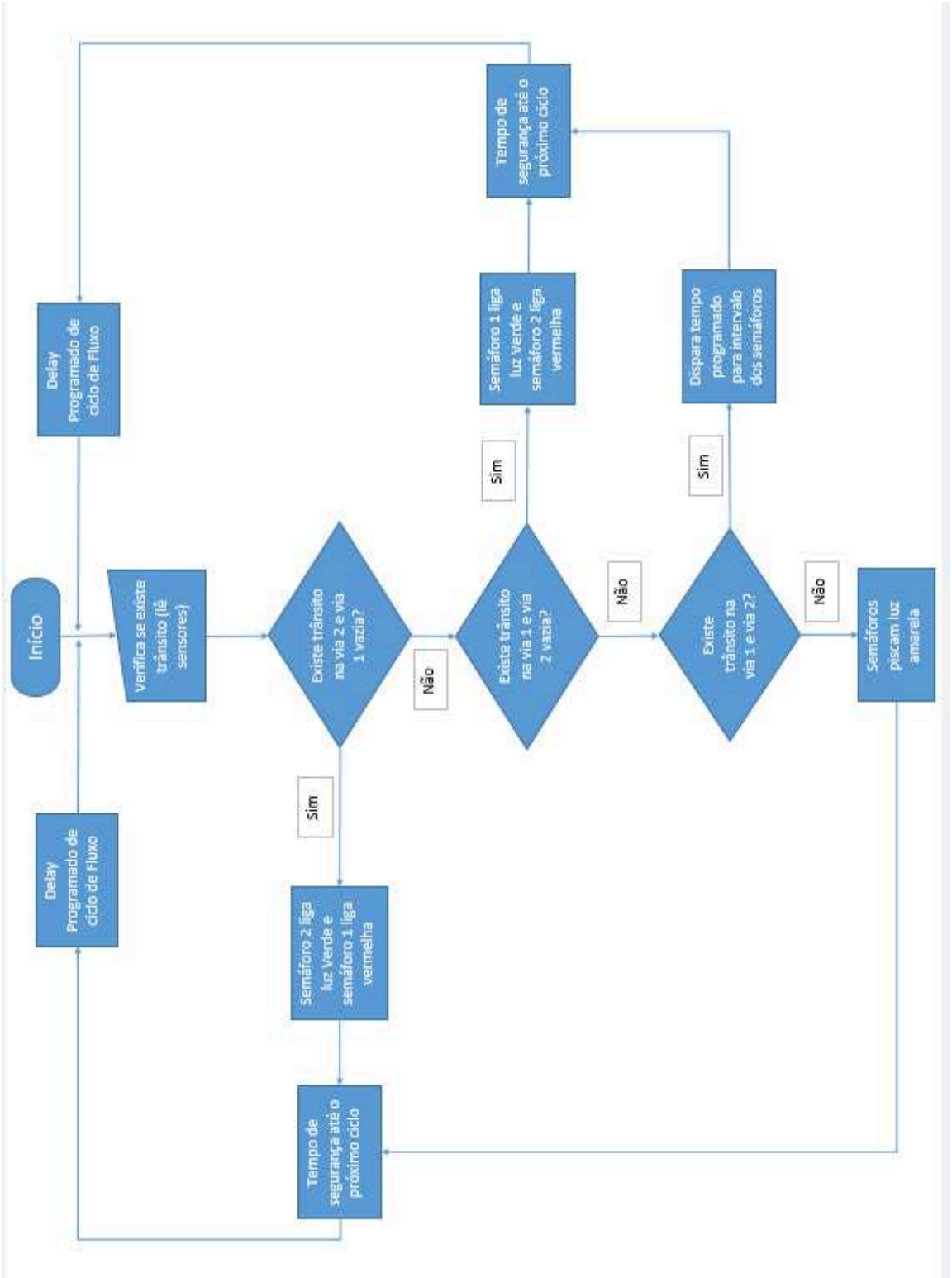
3.2 FUNCIONAMENTO

O funcionamento do sistema tem como finalidade evitar que veículos fiquem parados em faróis, desnecessariamente. O emprego de sensores infravermelhos faz com que o veículo seja detectado a uma distância segura do cruzamento, sendo que o sinal que o sensor emitir foi utilizado para informar ao microcontrolador qual o semáforo ele deve atuar.

No momento em que os dois sensores captarem a presença de veículos, em ambas as vias, o semáforo funciona de forma convencional, ou seja, temporizado.

A sequência do programa deve obedecer ao fluxograma gerado para o projeto que está ilustrado na figura 19.

Figura 19: Fluxograma completo



Fonte: Autor

O sistema funciona de forma cíclica, sendo que a cada condição que o microcontrolador encontrar, ele executa sua função e quando terminada, executa toda

varredura do programa novamente para verificar qual sua condição atual e assim por diante.

Foram conectados ao microcontrolador os LED's de cada semáforo e os sensores infravermelhos, e declarados como saídas digitais e entradas digitais respectivamente. O tempo de atuação dos LED's durante o funcionamento do projeto respeita o tempo mínimo de atuação de um semáforo comum de trânsito.

É possível se ter como base de funcionamento o circuito a partir da tabela da verdade, extraída através das condições que podem ser encontradas durante a aplicação pelos sinais dos sensores. A Tabela 6 apresenta a tabela verdade do processo em funcionamento.

Tabela 6: Tabela verdade do funcionamento do circuito

Sensor da via 1	Sensor da via 2	Semáforo da via 1	Semáforo da via 2
0	0	Amarelo intermitente	Amarelo Intermitente
0	1	Vermelho	Verde
1	0	Verde	Vermelho
1	1	Temporizado	Temporizado

Fonte: Autor

A tabela verdade tem como objetivo o emprego do sistema de falha segura, ou seja, quando houver os dois sensores atuados, o sistema opera em modo temporizado a fim de garantir a segurança no cruzamento, uma vez que todo sistema de detecção de presença possa ter equívocos na sua atuação devido a chuvas, ventos e tentativas de burlos.

4 RESULTADOS OBTIDOS

Com a montagem do circuito em uma placa didática, foi possível observar que os componentes utilizados para a aplicação atenderam a todos os objetivos do projeto. Foi utilizado um sensor infravermelho convencional para a simulação do projeto e adquirido o sensor recomendado JFL DSE-830, por possuir sensibilidade tripla e podendo evitar disparos equivocados, para a aplicação final do projeto.

Algumas dificuldades foram encontradas para implementar a programação na plataforma Arduino, fazendo com que fossem efetuadas algumas correções no programa para que a versão final atendesse às expectativas do projeto e pudesse ser empregado com segurança nas ruas e avenidas dos centros urbanos.

Na simulação, pôde-se verificar o que o sensor utilizado no projeto, HC-SR501, apresentou resultado esperado, podendo ser utilizado para detectar movimento de pequenos carros a fim de atuar o sistema de controle do microcontrolador, porém, este sensor possui algumas particularidades por não conseguir inibir disparos quando exposto a chuva, ventos, pequenos animais, o que não ocorreria no emprego do sensor proposto JFL.

O sistema final respondeu às expectativas, executando a programação corretamente, e o funcionamento dos componentes de forma correta e segura, que atendesse todos os quesitos do projeto.

5 CONCLUSÃO

Este trabalho, propõe como objetivo principal, o emprego de semáforos que são controlados por um sistema microcontrolador Arduino, e são conhecidos como semáforos inteligentes, que utilizam sensores infravermelhos para que, em horário de pouco movimento o veículo não fique parado, como hoje ocorre nos semáforos temporizados, evitando até riscos desnecessários ao motorista.

Para o seu desenvolvimento foi montado um protótipo em uma placa didática com o objetivo de simular o emprego destes sistemas nas vias urbanas de grandes centros, e seus resultados mostraram que o emprego deste sistema é eficiente e possibilita a melhora da qualidade do trânsito das cidades.

O grande desafio do sistema é a escolha de um sensor confiável que, na eventualidade de detecção equivocada devido, a chuvas, ventos ou burlas, o sistema opere com falha segura, garantindo a temporização das vias.

É possível inserir mais métodos de detecção que podem melhorar o projeto e poder controlar de forma mais eficiente o fluxo de veículos que circulam por uma via, portanto, como proposta para melhorias futuras, mantendo a busca por custo reduzido, é possível utilizar o emprego de sensores duplos, com algoritmos de detecção que possam evitar burlas ou equívocos no sensoriamento.

ANEXO 1 – PROGRAMAÇÃO BASEADA EM C++ DO PROJETO NO ARDUINO

```
const int sem1vermelho = 13;      // Led vermelho semáforo 1
const int sem1amarelo = 11;       // Led amarelo semáforo 1
const int sem1verde = 10;         // Led verde semáforo 1
const int sem2vermelho = 9;       // Led vermelho semáforo 2
const int sem2amarelo = 8;        // Led amarelo semáforo 2
const int sem2verde = 7;          // Led verde semáforo 2
int sensor1 = 6;                  // Define sensor infravermelho via 1
int sensor2 = 3;                  // Define sensor infravermelho via 2

void setup(){
      pinMode(sem1vermelho, OUTPUT);      // Define característica do pino
      pinMode(sem1amarelo, OUTPUT);       // Define característica do pino
      pinMode(sem1verde, OUTPUT);         // Define característica do pino
      pinMode(sem2vermelho, OUTPUT);      // Define característica do pino
      pinMode(sem2amarelo, OUTPUT);       // Define característica do pino
      pinMode(sem2verde, OUTPUT);         // Define característica do pino
      pinMode(sensor1, INPUT);            // Define característica do pino
      pinMode(sensor2, INPUT);            // Define característica do pino
}
void loop() {
      int state1 = digitalRead(sensor1);  // Verifica estado do sensor 1
      int state2 = digitalRead(sensor2); // Verifica estado do sensor 2

if (state1 == HIGH && state2 == LOW){
// Desliga as lâmpadas de pare e siga de ambos semáforos
```

```
            digitalWrite(sem1vermelho, LOW);
            digitalWrite(sem2vermelho, LOW);
            digitalWrite(sem1verde, LOW);
            digitalWrite(sem2verde, LOW);
// Liga lâmpadas de atenção em ambos os semáforos durante 4 segundos
            digitalWrite(sem1amarelo, HIGH);
            digitalWrite(sem2amarelo, HIGH);
            delay (4000);
            digitalWrite(sem1amarelo, LOW);
            digitalWrite(sem2amarelo, LOW);
// Abre semáforo 1 durante 15 segundos para que o carro passe em segurança
            digitalWrite(sem1verde, HIGH);
// Fecha o semáforo 2 para que o trânsito flua na outra via
            digitalWrite(sem2vermelho, HIGH);
            delay (15000);
      }
      if (state1 == LOW && state2 == HIGH){
            digitalWrite(sem1vermelho, LOW);
            digitalWrite(sem2vermelho, LOW);
            digitalWrite(sem1verde, LOW);
            digitalWrite(sem2verde, LOW);
            digitalWrite(sem1amarelo, HIGH);
            digitalWrite(sem2amarelo, HIGH);
            delay (4000);
            digitalWrite(sem1amarelo, LOW);
```

```
            digitalWrite(sem2amarelo, LOW);
            digitalWrite(sem2verde, HIGH);
            digitalWrite(sem1vermelho, HIGH);
            delay (15000);
        }
        if (state1 == HIGH && state2 == HIGH){
// Quando houver veículos em ambas as vias
            digitalWrite(sem1vermelho, LOW);
            digitalWrite(sem2vermelho, LOW);
            digitalWrite(sem1verde, LOW);
            digitalWrite(sem2verde, LOW);
            digitalWrite(sem1amarelo, HIGH);
            digitalWrite(sem2amarelo, HIGH);
            delay (4000);
            digitalWrite(sem1amarelo, LOW);
            digitalWrite(sem2amarelo, LOW);

// Enquanto ambos os sensores permanecerem altos os semáforos operam em
modo convencional
            if (state1 == HIGH && state2 == HIGH){
                digitalWrite(sem2amarelo, LOW);
//apaga o led vermelho (sinal 1)
                digitalWrite(sem1vermelho, LOW);
//acende o led verde (sinal 1)
                digitalWrite(sem1verde, HIGH);
//acende o led vermelho (sinal 2)
                digitalWrite(sem2vermelho, HIGH);
```

```
//espera 15 segundos
                delay (15000);
//apaga o led verde (sinal 1)
                digitalWrite(sem1verde, LOW);
//acende o led amarelo (sinal 1)
                digitalWrite(sem1amarelo, HIGH);
//espera 4 segundos
                delay (4000);
//apaga o led amarelo (sinal 1)
                digitalWrite(sem1amarelo, LOW);
//acende o led vermelho (sinal 1)
                digitalWrite(sem1vermelho, HIGH);
//apaga o led vermelho (sinal 2)
                digitalWrite(sem2vermelho, LOW);
//acende o led verde (sinal 2)
                digitalWrite(sem2verde, HIGH);
//espera 15 segundos
                delay (15000);
//apaga o led verde (sinal 2)
                digitalWrite(sem2verde, LOW);
//acende o led amarelo (sinal 2)
                digitalWrite(sem2amarelo, HIGH);
//espera 4 segundos
                delay(4000);
//apaga o led amarelo (sinal 2)
```

```
                digitalWrite(sem2vermelho, LOW);
            }
        }

        if (state1 == LOW && state2 == LOW){

// Verifica o estado dos sensores e cria um loop While enquanto os sensores
permanecerem desligados

            digitalWrite(sem1vermelho, LOW);

            digitalWrite(sem2vermelho, LOW);

            digitalWrite(sem1verde, LOW);

            digitalWrite(sem2verde, LOW);

            digitalWrite(sem1amarelo, HIGH);

            digitalWrite(sem2amarelo, HIGH);

            delay (500);

            digitalWrite(sem1amarelo, LOW);

            digitalWrite(sem2amarelo, LOW);

            delay (500);
        }
}
```

REFERÊNCIAS

ARANTES, R. A., **Controle de Semáforo Eletrônico Inteligente**, Trabalho de Conclusão de Curso, Universidade de Taubaté, SP - 2010 – 69 Páginas.

ARAÚJO, I. B. Q. et al. **Desenvolvimento de um protótipo de automação predial/residencial utilizando a plataforma de prototipagem eletrônica arduino**. COBENGE 2012 - XL Congresso Brasileiro de Educação em Engenharia, Belém, PA, set. 2012. Disponível em: <http://www.abenge.org.br/CobengeAnteriores/2012/artigos/103723.pdf>. Acesso em: 30 de agosto de 2017.

ASSIS, P. D. K. B., **MICROCONTROLADOR**. Universidade Presidente Antônio Carlos – Faculdade de Ciência da Computação e Comunicação Social, 2004. Disponível em: < http://www.unipac.br/site/bb/tcc/tcc-f6cceedfa3f6307211208b80c790c6e3.pdf> Acesso em 23 de outubro de 2017.

BERTOGNA, E. G., **Microcontrolador AVR Baseado no ATMega328**, 2013. Universidade Tecnológica Federal do Paraná, Disponível em: <http://paginapessoal.utfpr.edu.br/ebertonha/apostilas/el83e/Slides_ATmega328.PDF/at_download/file>. Acesso em 29 de agosto de 2017.

BRAGA, C. N., **Como funciona o Cristal na Eletrônica ART.423 – Cristal de Quartzo**, 2017. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/3081-art423> Acesso em 02 de setembro de 2017.

CAETANO, R. **ATMega328 – Microcontrolador Atmel ATMega328**. 2007. Disponível em: <https://sites.google.com/site/ronaldoecaetano/microcontrolador/atmega328> Acesso em: 28 de agosto de 2017.

CASAVELLA, E., **O que é Linguagem C?**. Artigo online. Disponível em: <http://linguagemc.com.br/o-que-e-linguagem-c/> Acesso em: 30 de agosto de 2017.

FERREIRA, E. A. **Aplicação da plataforma Arduino para a determinação de parâmetros atmosféricos e ambientais**. CONIC-SEMESP, 15º congresso Nacional de Iniciação Científica, 2015. Disponível em: <http://conic-semesp.org.br/anais/files/2015/trabalho-1000020063.pdf>. Acesso em: 11 de junho de 2017.

FILHO, D. O. B., **Construindo sua placa Arduino**. Curso de Arduino, artigo online. 2012. Disponível em: < http://www.robotizando.com.br/curso_arduino_construindo_pg1.php> Acesso em: 11 de novembro de 2017.

HOEPERS, R., **Veículo autônomo utilizando Arduino**. Universidade do Vale do Itajaí – Centro de Ciências Tecnológicas da Terra e do Mar – Ciência da Computação,

2012. Disponível em: <http://siaibib01.univali.br/pdf/Rodrigo%20Hoepers.pdf> Acesso em: 24 de out de 2017.

JFL, DSE 830. **Manual do sensor infravermelho micro-ondas duplo DSE830**. 2017. JFL Alarmes e tecnologia em segurança. Disponível em: <http://jflalarmes.tecnologia.ws/uploads/jfl-download-passivos-manual-dse-830-.pdf> Acesso em: 28 de julho de 2017.

JFL, IRD 640. **Manual do sensor infravermelho duplo IRD640**, 2007. JFL Alarmes e tecnologia em segurança. Disponível em: <http://jflalarmes.tecnologia.ws/uploads/jfl-download-passivos-ird-640-pet.pdf> Acesso em: 13 de novembro de 2017.

MATTOS, E. A.; MANTELI, R. V., **Semáforo Inteligente de Trânsito**, Trabalho de Conclusão de Curso, Universidade de Taubaté – SP, 2006. 65 Páginas.

MAZZAROPPI, M., **Sensores de movimento e presença**. 2007, Universidade Federal do Rio de Janeiro, Escola Politécnica, Departamento de Engenharia Elétrica. Disponível em : <http://monografias.poli.ufrj.br/monografias/monopoli10001369.pdf> Acesso em: 13 de novembro de 2017.

MCROBERTS, M., **Arduino Básico**, 2011. Disponível em: < http://adjutojunior.com.br/arduino/arduino_b%C3%A1sico_Michael_McRoberts.pdf> Acesso em 11 de novembro de 2017.

MÉLO, F. E. N. et al. **Do scratch ao Arduino: uma proposta para o ensino introdutório de programação para cursos superiores de tecnologia**. COBENGE 2011 – XXXIX Congresso Brasileiro de Educação em Engenharia, Blumenau, SC, out. 2011Disponível em: <http://www3.fsa.br/LocalUser/cobenge2011/sessoestec/art1886.pdf>. Acesso em: 13 agosto de 2017.

NETO, B. B. O.; MONTEIRO, P. F.; QUEIROGA, S. L. M. **Aplicabilidade dos Microcontroladores em Inovações Tecnológicas**, 2012. Disponível em: <http://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/2433/2526>. Acesso em 22 agosto 2017.

OLIVEIRA, S. F.; MARCELINO, M. A.; PRADO, P. P. L., et al. **Semáforo Inteligente de baixo custo com sistema de detecção segura**. Revista SODEBRAS – Volume 8 – Nº 85, Janeiro, 2013 - Universidade de Taubaté – Brasil. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Brasil. Universidade Estadual Paulista.

PALHETA, F. C., Ciência, Tecnologia e Sociedade. Universidade Federal do Pará. **Radiação Infravermelha**. 2012 Disponível em: <http://www.ufpa.br/ensinofts/radiologia.html> Acesso em: 03 de setembro de 2017.

PINTO, P. **Mundo Arduino: Vamos começar a programar?**. 2014. Disponível em: <https://pplware.sapo.pt/gadgets/hardware/mundo-arduino-vamos-comecar-a-programar/.> Acesso em: 26 de agosto de 2017.

SILVEIRA, J. A., **Arduino - Cartilha para programação em C**. Edição janeiro de 2012. Disponível em: <http://ordemnatural.com.br/pdf-files/CartilhadoArduino_ed1.pdf> Acesso em 31 de agosto de 2017.