SINCRONIZAÇÃO ENTRE SEMÁFOROS: SIMULAÇÃO DE ONDA VERDE UTILIZANDO A PLATAFORMA ARDUINO

João Paulo Amaral Melo[1]
Gabriel Pereira Santos[1]
José Islan Rodrigues Galvão[1]
Cainã Sena da Silva[1]
José Abrain Alves Nascimento[1]

Resumo: Os semáforos são implementados com o objetivo de disciplinar o movimento das correntes de tráfego, mas apesar disso, as sinalizações podem gerar desconforto e problemas. O objeto geral deste trabalho é buscar e apresentar uma forma de minimizar os efeitos negativos causados pela sinalização semafórica, fazendo uso da ferramenta Arduino como plataforma para o desenvolvimento desse sistema.

Palavras-chave: Semáforo. Sincronização Semafórica. Arduino.

Abstract: Traffic lights are implemented in order to discipline the movement of traffic streams, but despite this, the signs can generate discomfort and problems. The general objective of this work is to find and present a way to minimize the negative effects caused by traffic light signaling, making use of the Arduino tool as a platform for the development of this system.

Keywords: Semaphore. Semaphoretic Synchronization. Arduino.

1. INTRODUÇÃO

Os semáforos são implementados com o objetivo de disciplinar o movimento das correntes de tráfego, controlando a passagem de vias concorrentes e de pedestres, reduzindo assim a probabilidade de eventuais acidentes e promovendo o fluxo contínuo de veículos. “O semáforo é um dispositivo de controle de tráfego que, através de indicações luminosas transmitidas para motoristas e pedestres, alterna o direito de passagem de veículos e/ou pedestres em interseções de duas ou mais vias.” (DENATRAN, 1984, p.14). Apesar disso, as sinalizações podem gerar desconforto e problemas, como por exemplo, o aumento do percurso, formação de enormes filas e a diminuição da velocidade média.

[1]UNIVERSIDADE GUARULHOS - UNG. Acadêmicos do curso de Ciência da Computação. Contato do autor responsável: jose-islan52@hotmail.com.

O trânsito brasileiro vem se tornando uma das maiores problemáticas dentro dos grandes centros urbanos e dentre as principais causas está, sem dúvida, o intenso congestionamento de veículos, contribuindo para um elevado nível de estresse dos cidadãos. Um dos muitos motivos está na má construção semafórica, onde os semáforos distribuídos possuem quase nenhuma inteligência. Uma pesquisa sobre mobilidade urbana realizada pelo Ibope constatou que o paulistano gasta, em média, um mês e meio parado no trânsito no ano. No ano de 2015, os paulistanos passavam quase duas horas e quarenta minutos presos em congestionamentos, tempo este que subiu para quase três horas no ano seguinte.

Em vista às complicações no trânsito envolvendo os semáforos, este trabalho apresenta uma simulação de um sistema de sincronização semafórica, a chamada “Onda Verde”, como forma de minimizar os problemas apresentados, fazendo uso da ferramenta Arduino como plataforma para o desenvolvimento e desse sistema.

O objeto geral deste trabalho é buscar e apresentar, através de uma simulação em maquete, uma forma de minimizar os efeitos negativos causados pela sinalização semafórica, através de um sistema de sincronização entre semáforos, procurando contribuir para uma melhoria na situação atual do trânsito.

2. FUNDAMENTAÇÃO TEÓRICA

Segundo o DENATRAN (1984) o semáforo é um dispositivo que serve para controlar o tráfego de veículos e pedestres em uma via, através de indicações luminosas, geralmente fixados entre os cruzamentos das mesmas. Sendo assim, existem dois tipos de semáforos: os de veículos (veiculares) e os de pedestres.

O semáforo veicular é geralmente composto por três cores (focos de luz), onde cada uma delas tem um significado. Tais cores são regidas por padrões internacionais. São elas:

Cor verde: indica que os veículos podem seguir em frente;

Cor amarela: indica que os veículos devem parar antes de entrar no cruzamento. Caso contrário, os veículos devem seguir em frente sem que haja risco para a segurança do tráfego;

Cor vermelha: indicam que os veículos devem parar e permanecer parados até que haja autorização para prosseguir.

O semáforo de pedestre também segue o mesmo raciocínio, contendo

geralmente dois focos de luz; um na cor verde, que significa que o pedestre pode atravessar e outro na cor vermelha, que significa que o pedestre é impedido de atravessar até que seja autorizado.

De acordo com NINACANSAYA (2016) os semáforos são classificados em:

a) semáforos pré-sincronizados e de tempo pré-determinado;

b) semáforos acionados pelo tráfego do trânsito;

c) semáforos com controle centralizado.

As mudanças de estado de um semáforo são feitas a partir de um controlador de tráfego. É ele quem ordena quando o semáforo pode mudar de uma cor para outra, através do envio de pulsos elétricos. No que diz respeito ao tempo de envio dos pulsos elétricos, são determinados de duas maneiras. DENATRAN (1984, p.26):

> a) **manual** - os comandos de verde/amarelo/vermelho são acionados de forma manual, geralmente por um guarda de trânsito. Neste tipo de operação, a duração dos estágios obedece a critérios pessoais de julgamento da situação de tráfego e, normalmente, nem um ciclo nem o tempo de verde das fases são constantes ao longo do tempo;
> b) **automática** - o tempo do ciclo, duração e instantes de mudança dos estágios são definidos pelo controlador, através de uma programação interna, cuja lógica tanto pode ser bastante simples como sofisticada, dependendo do tipo de controlador em questão.

Foi escolhida a plataforma Arduino para o desenvolvimento deste trabalho. É uma plataforma de prototipagem em eletrônica popular que começou como um projeto de pesquisa, isso no início dos anos 2000. Através dessa plataforma é possível ter controle de diversos dispositivos como sensores e motores, por exemplo, facilitando na construção tanto de projetos simples até os mais robustos e complexos. NINACANSAYA (2016) define Arduino como uma companhia de tecnologia *open source* que confecciona placas eletrônicas, onde é possível a integração de um microcontrolador presente em cada placa fabricada e uma IDE (ambiente de desenvolvimento de software).

O ambiente de desenvolvimento de software (IDE) para integração com o microcontrolador presente na placa impressa se chama Arduino IDE. “O software de desenvolvimento Arduino é bastante fácil e intuitivo de utilizar, não havendo qualquer nível de dificuldade.” SANTOS (2009, p.15). A integração entre o código desenvolvido e o hardware é feito através de uma comunicação serial (USB).

3. MATERIAIS E MÉTODOS

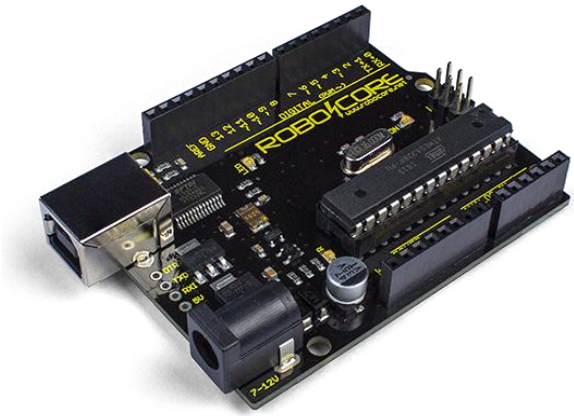
3.1 BlackBoard V1.0 - RoboCore

Para o desenvolvimento deste projeto, foi utilizado a placa BlackBoard V1.0, desenvolvido pela RoboCore. Como a tecnologia Arduino (hardware e software) é baseada na filosofia *open source*, diversas outras placas não genuínas, conhecidas como placas-clone, são desenvolvidas no mercado, sendo 100% compatíveis. O hardware usado neste projeto é um exemplo de uma placa-clone. Isso é melhor explicado por MCROBERTS (2011, p.24):

> "O hardware e o software do Arduino são ambos de fonte aberta, o que significa que o código, os esquemas, o projeto etc. podem ser utilizados livremente por qualquer pessoa e com qualquer propósito. Dessa forma, há muitas placas-clone e outras placas com base no Arduino disponíveis para compra, ou que podem ser criadas a partir de um diagrama."

Para que a placa seja corretamente reconhecida pelo seu computador, é necessária a instalação do driver FTDI[2].

Figura 1 - placa BlackBoard



Fonte: Internet.

Esta placa possui um microcontrolador ATmega328, o qual possui o *bootloader* do Arduino Uno R3. Sendo assim, ao se conectar à IDE, é preciso selecionar a placa do Arduino Uno R3. Opera numa tensão de 5V, possui uma memória flash de 32 KB;

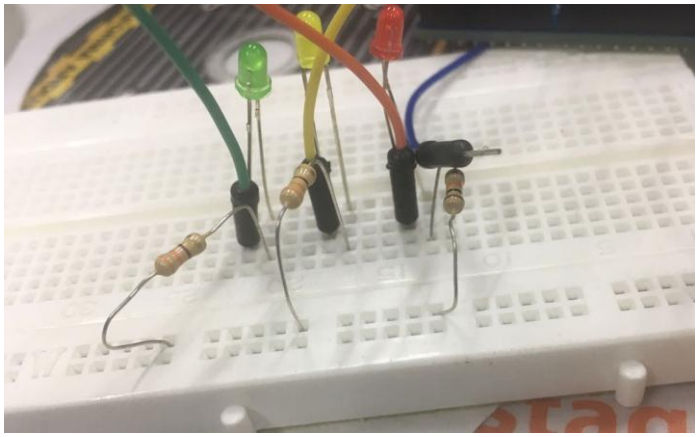
[2] FTDI é o driver responsável por fazer a conversão do sinal serial da porta USB para o microcontrolador da placa. Placas Arduino, com o lançamento da versão Uno, não necessitam da instalação desse driver, uma vez que utilizam microcontroladores que fazem essa conversão internamente, além de ser programado com um firmware que, ao se conectar com um computador, cria a porta de comunicação serial. (www.robocore.net).

onde 0,5 KB dela são usados para a gravação do *bootloader*, uma memória SRAM de 2 KB e uma EEPROM de 1 KB.

3.2 Dispositivos eletrônicos e outros materiais

Usamos de resistores e LEDs para simular a operação de semáforos dentro da nossa maquete. Os resistores usados foram os com uma impedância de 220 ohms, pelo fato de ser o mais confiável para a estrutura do projeto. Usamos de placas *protoboards* de 840 pinos para a conexão dos LEDs junto à programação desenvolvida.

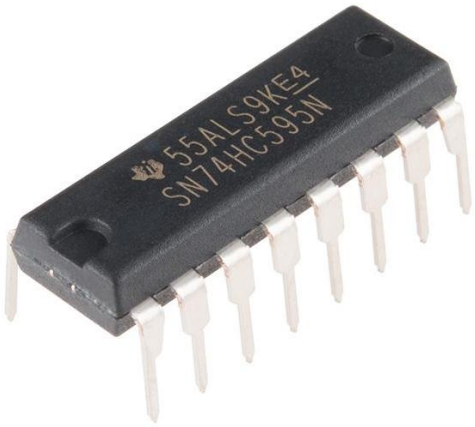
Figura 2 - LEDs e resistores integrados à *protoboard*



Fonte: Autor.

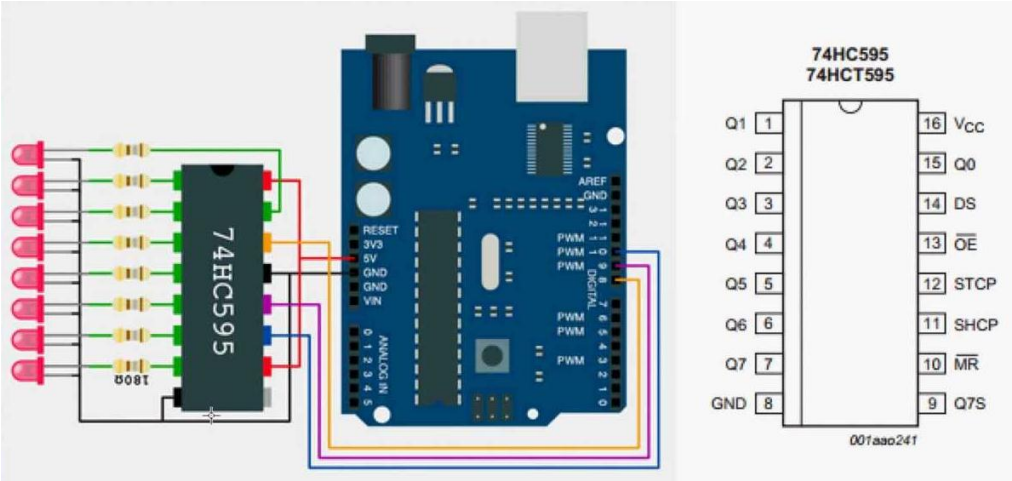
Além destes, também foi usado CIs (Circuito Integrado) do tipo *shift register* na implementação do trabalho. O modelo usado foi o 74HC595. Esse dispositivo é usado com a finalidade aumentar o número de pinagens de saída do hardware. Em resumo, esse chip possui 16 pinos, onde 3 são destinados à comunicação com a placa (pinos 11,12 e 14) e dispõe de 8 pinos de saída (pinos 15, 1, 2, 3, 4, 5, 6 e 7).

Figura 3 - Circuito Integrado modelo 74HC595



Fonte: Internet.

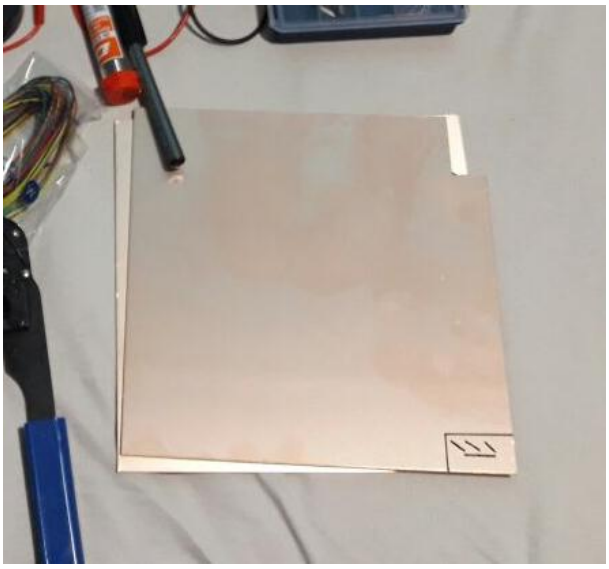
Figura 4 - Funcionamento do CI



Fonte: Internet.

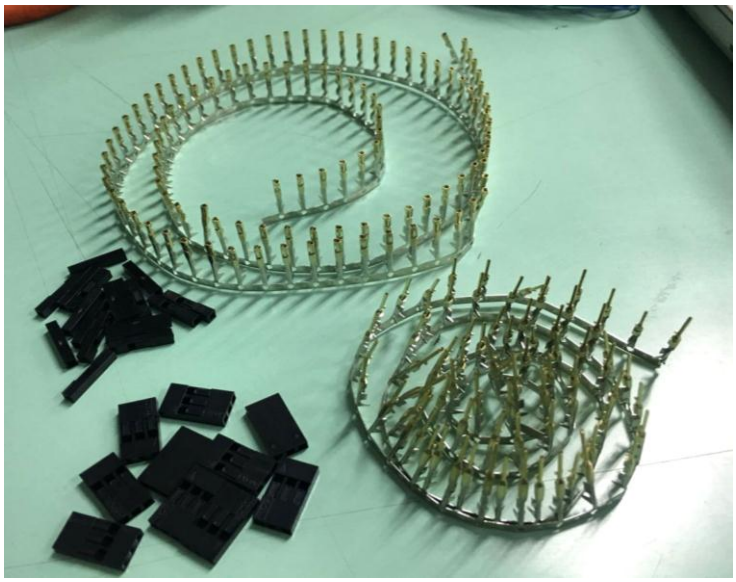
No que diz respeito à confecção dos semáforos, tanto os veiculares quanto os de pedestres, utilizamos canos de mangueira de polietileno de 4.6 mm (3/16”) para o desenvolvimento dos postes; placa de fenolite, onde os LEDs e resistores serão soldados para a representação das caixas dos semáforos e de cabos de rede (par trançado) para fazer a comunicação dos semáforos. Para a conexão da fiação dos semáforos com a placa *protoboard*, foi preciso de terminais jumper (macho e fêmea).

Figura 5 - Placa de fenolite



Fonte: Autor.

Figura 6 - Terminais jumper (macho e fêmea) e capas de 1 e 3 pinos



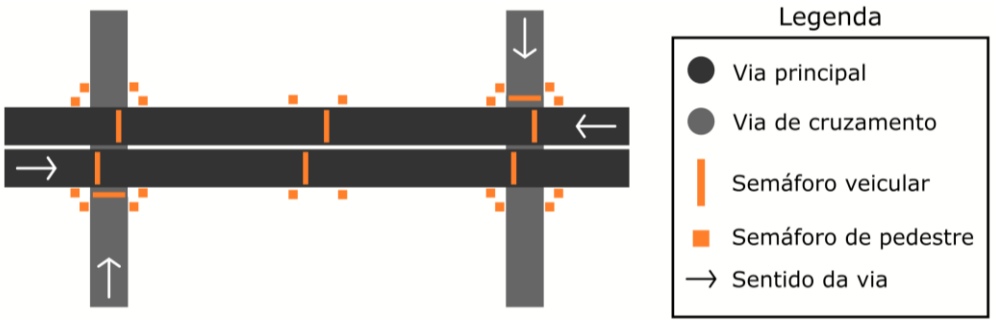
Fonte: Autor.

3.3. Estrutura de vias e sincronização

A ideia da nossa estrutura de sincronia semafórica foi aplicá-la em uma via de mão dupla (mão e contramão). Os semáforos foram distribuídos através de blocos; cada bloco é composto de três semáforos e dois cruzamentos. Na representação da maquete foi implementado apenas um bloco. Com isso, a nossa representação dispõe de oito semáforos veiculares; sendo seis distribuídos na via principal e um

em cada via que fará o cruzamento com a principal, e vinte semáforos de pedestres; sendo oito distribuídos em cada um dos cruzamentos e dois no farol central do bloco.

Figura 7 - Esquema da estrutura de vias e disposição dos semáforos da maquete

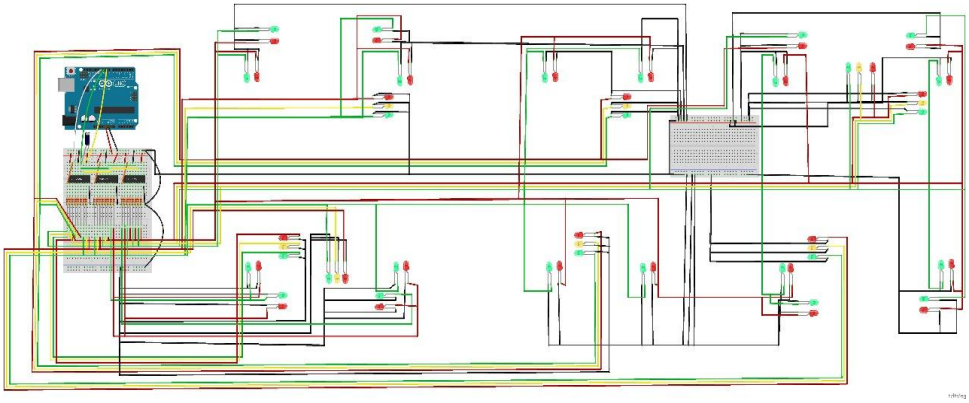


Fonte: Autor

3.4. Estrutura da maquete

A base da maquete foi desenvolvida em madeira MDF. Nela foi implementado um conjunto de fios que farão a comunicação dos semáforos com a placa de prototipagem usada. Cada ponto onde ficará fixado um semáforo foram postos terminais jumper (fêmea).

Figura 8 - Esquema da estrutura de fiação



Fonte: Autor.

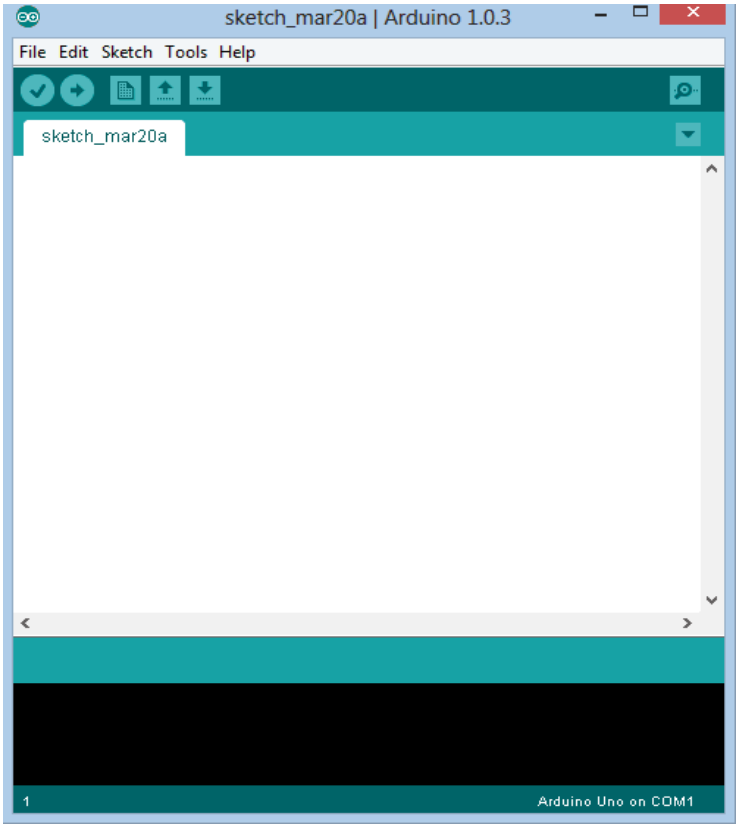
3.5. Plataforma IDE

Como dito anteriormente, o software responsável pelo desenvolvimento da

programação (IDE) que será integrada junto à placa é o Arduino IDE[3]. Nele foi desenvolvido o código referente à rotina de atuação dos semáforos da maquete como também a configuração dos CIs usados. Após tudo isso, foi feito o *upload* do código para o hardware via comunicação serial.

A interface da IDE é bem intuitiva e de fácil compreensão. O código dentro dela desenvolvido é chamado de *sketch*.

Figura 9 - Interface do Arduino IDE



Fonte: Internet.

A interface da IDE é basicamente dividida em três partes: uma barra de ferramentas na parte superior, a área de escrita do código no centro e uma área de exibição de mensagens na parte inferior da IDE. Na barra de ferramentas (ou *Toolbar*) é possível navegar por guias com diferentes códigos que estejam sendo desenvolvidos. Acima da(s) guia(s) há algumas opções de men u. São elas: Arquivo (ou *File*); Editar (ou *Edit*); *Sketch*; Ferramentas (ou *Tools*) e Ajuda (ou *Help*). Abaixo

[3] O software está disponível para *download* de forma gratuita no próprio site do Arduino (https://www.arduino.cc/en/Main/software).

dessas opções há um menu de ícones que servem de atalho para as funções mais requisitadas. São ícones de verificação de erro no código (*Verify*); compilação e gravação do código na placa (*Upload*); criação de uma aba para a escrita de um novo código (*New*); abertura de um *sketch* previamente criado e salvo (*Open*); gravação do código da guia que está ativa ou em uso (*Save*) e a abertura do monitor serial (*Serial Monitor*). Dentro do código desenvolvido existem duas funções obrigatórias: a função setup ( ); que é executada no início do programa, responsável pelas configurações iniciais do microcontrolador e a função loop ( ); onde é inserido todo o código referente à programação do microcontrolador, ou seja, é um laço que será continuamente executado.

4. RESULTADOS E DISCUSSÕES

Para usarmos a placa de fenolite como placa de circuito impresso, usamos do produto percloreto de ferro; que é um produto altamente corrosivo, necessário para a remoção dos resíduos de cobre presente na placa. Antes de fazer essa remoção com o produto, foi desenhado o circuito com tinta permanente (caneta de CD) na placa de fenolite para posteriormente fazer a fixação dos componentes eletrônicos.

Figura 10 - Imersão da placa de fenolite no percloreto de ferro

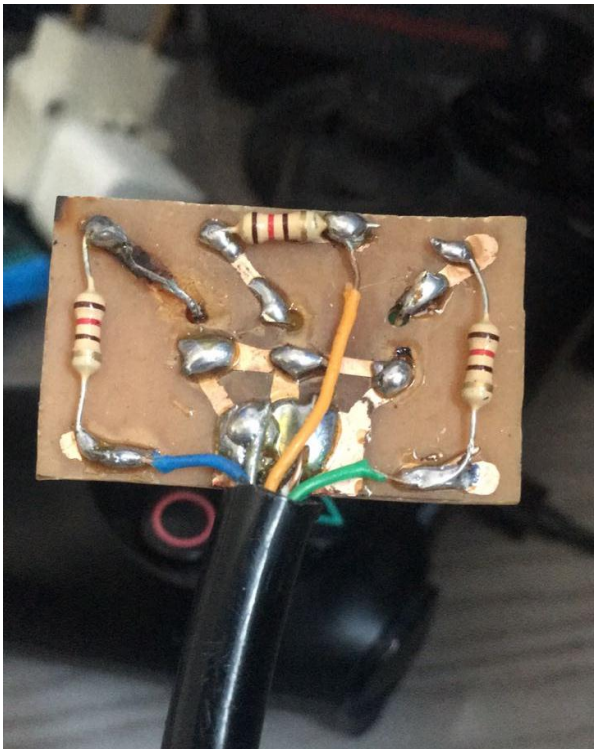
Fonte: Autor.

Figura 11 - Limpeza da placa após lavagem com água corrente

Fonte: Autor.

A confecção das caixas semafóricas foi feita com uso dos LEDs vermelho, verde e amarelo; representando os focos de luz, onde foram soldados juntamente com os resistores na placa de fenolite.

Figura 12 - Montagem das caixas dos semáforos



Fonte: Autor.

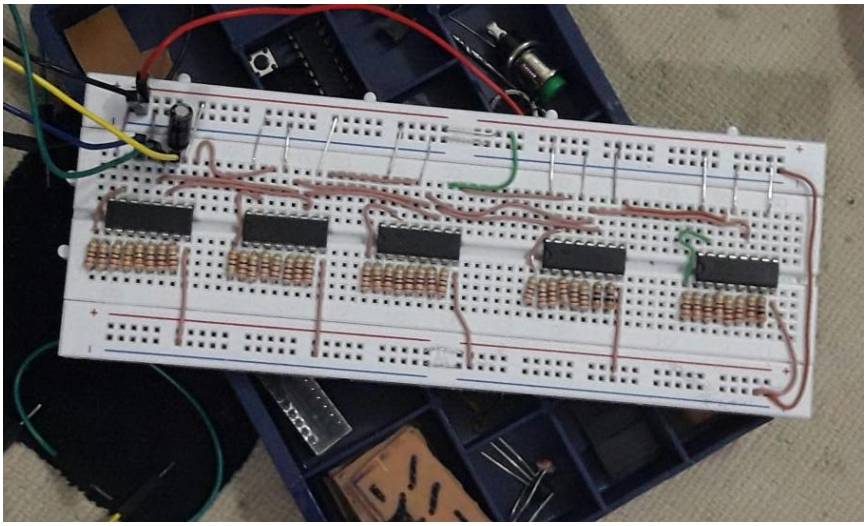
Para cada poste veicular foi utilizado 4 pinos (ou terminais) jumper macho; três referente aos fios ligados na polaridade positiva de cada um dos LEDs (vermelho, amarelo e verde) e um referente ao fio ligado na polaridade negativa dos LEDs, sendo este útlimo único igual para todos os LEDs do poste confeccionado. O mesmo aconteceu com os postes de pedestres, sendo utilizado de 3 pinos jumper macho; dois referente aos fios ligados na polaridade positiva de cada um dos LEDs (vermelho e verde) e um referente ao fio ligado na polaridade negativa dos LEDs. A pinagem presente em cada poste semafórico (terminais jumper macho) é que fará a ligação com pinagem existente na estrutura de fiação da maquete (terminais jumper fêmea).

Figura 13 - Confecção dos postes semafóricos

Fonte: Autor.

O uso do circuito integrado (CI) tornou possível a expansão das portas lógicas da placa de prototipagem e, com isso, foi possível o controle independente de cada LED presente nos postes semafóricos. Sendo assim, foram utilizadas 3 portas lógicas da placa de prototipagem para comunicação exclusiva com o CI. Foram utilizados de 5 CIs, todos ligados em série; o que nos permitiu tornar uma mesma comunicação com a placa única para todos, e assim nos proporcionando a quantidade necessária de portas lógicas para o desenvolvimento do projeto.

Figura 14 - Montagem dos CIs para a expansão das portas lógicas



Fonte: Autor.

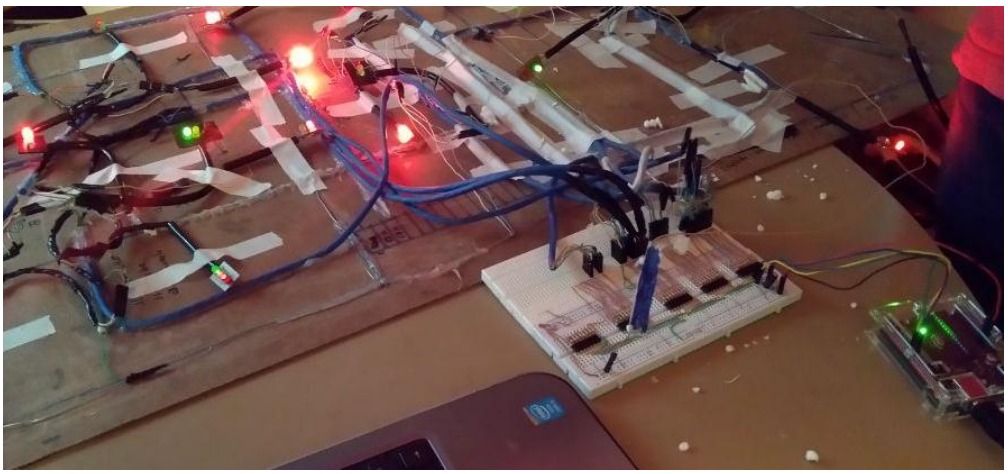
Foi desenvolvido uma interface gráfica em linguagem Java com o propósito de o usuário fazer a seleção do modo de atuação dos semáforos da maquete; seja o modo de sincronização semafórica, modo normal ou o modo piscante. Dentro do código desenvolvido foi feita a comunicação com a porta serial referente à placa de prototipagem. Desse modo, os dados selecionados na interface são enviados para a placa através dessa comunicação estabelecida dentro do código Java.

Figura 15 - Interface de controle do modo de atuação da maquete

Fonte: Autor.

Após a finalização de todas essas etapas, a maquete foi montada seguindo o modelo de vias apresentado no trabalho. Sobre a base usada foi posto estruturas feitas de isopor, tanto para encobrir a fiação da maquete quanto para servir de suporte para a simulação das vias. Além disso, essa estrutura de isopor ajudou na melhor fixação dos postes semafóricos junto à maquete.

Figura 16 – Estrutura de fiação da maquete



Fonte: Autor.

Figura 17 - Resultado final da maquete

Fonte: Autor.

5. CONCLUSÃO

Uma das principais dificuldades dentro da ideia de sincronização semafórica é implantá-la em uma via de mão dupla (mão e contramão) justamente pelo sentido oposto que uma mão tem para com a outra; o sentido da sincronização semafórica de uma mão da via será o sentido oposto da sincronização da mão contrária a ela.

O foco principal deste trabalho foi pensar em um modelo que atendesse a essa problemática e representá-lo no desenvolvimento de uma maquete juntamente com a utilização da plataforma Arduino. Com isso, não nos preocupamos em definir valores reais para dimensões das vias e/ou velocidade média das mesmas.

Dessa forma, o modelo utilizado neste trabalho se mostrou eficaz para a utilização da sincronização semafórica, sendo o modelo composto de uma via principal de mão dupla com a disposição de três semáforos veiculares e dois cruzamentos.

6. REFERÊNCIAS

BANZI, M.; SHILOH, M. **Primeiros Passos com o Arduino**. Tradução de Aldir José Coelho Corrêa da Silva. 2ª ed. São Paulo: Novatec Editora, 2015. 240 p.

BRASIL. Departamento Nacional de Trânsito. **Manual de Semáforos**. 2ª ed. Brasília: DENATRAN, 1984. 172 p. (Col. serviços de engenharia, 4).

MCROBERTS, Michael. **Arduino Básico.** Tradução de Rafael Zanolli. São Paulo: Novatec Editora, 2011. 456 p.

NINACANSAYA, A. R. M. **Analisis y diseño de un sistema de control de trafico vehicular utilizando semaforos inteligentes con tecnología Arduino**. 2016. 79 p. Tesis (Título Profesional de Ingeniero Electrónico). Universidad Nacional del Altiplano Puno, Facultad de Ingeniería Mecánica Eléctrica, Electrónica y Sistemas. Puno, 2016.

KRAUSS, Mauricio. **Automação de Sistema Semafórico**. 2014. 67 p. Trabalho de Conclusão de Curso - Tecnologia em Automação Industrial, UTFPR - Universidade Tecnológica Federal do Paraná. Curitiba, 2014. Disponível em:<http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/3145/1/CT_COALT_2014_1_11.pdf>. Acesso em: 16 set. 2017.

SANTOS, Nuno Pessanha. **Arduino - Introdução e Recursos Avançados**. 2009, 69 p. Departamento de Engenheiros Navais, Escola Naval, 2009. Disponível em:<http://www.novaims.unl.pt/docentes/vlobo/escola_naval/MFC/Tutorial%20Arduino.pdf>. Acesso em: 14 out. 2017.

SMIDT, A. C. G. **Implementação de uma Plataforma Robótica controlada remotamente utilizando Arduino**. 2013, 82 p. Monografia (Graduação em Engenharia de Computação), Escola de Engenharia de São Carlos da Universidade de São Paulo, São Carlos, 2013. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis/97/970010/tce-19112013-093717/?&lang=br>. Acesso em: 8 out. 2017.