UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

Montagem de uma bicicleta elétrica controlada e monitorada através de um aplicativo Android

Dayse Maranhão Cavalcanti

Orientador: Prof. Dr. Samaherni Morais Dias

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia Mecatrônica da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do título de Engenheiro Mecatrônico, orientado pelo Prof. Dr. Samaherni Morais Dias.

Natal - RN
2016

Montagem de uma bicicleta elétrica controlada e monitorada através de um aplicativo Android

Dayse Maranhão Cavalcanti

Aprovado em ____ de _____ de 2016.

Prof. Dr. Samaherni Morais Dias (orientador) . . . . . DEE/UFRN

Prof. Dr. Kurios Iuri Pinheiro de Melo Queiroz . . . . . DEE/UFRN

Prof. Dr. Joilson Batista de Almeida Rego . . . . . ECT/UFFN

Agradecimentos

Ao meu orientador, sou grata pela disposição em participar deste projeto e por me mostrar qual caminho seguir.

Ao meu soldador e mestre das gambiarras, agradeço pela ajuda na montagem desse quebra-cabeça.

À minha maior benfeitora e protetora, sou mais do que grata pelo cuidado, dedicação e carinho.

Às minhas duas amadas e aos seus pimpolhos, por me alegrarem nos momentos difíceis e me darem esperança.

Ao meu professor particular e melhor amigo, por todos os afagos que seguiam as patadas e puxões de orelha.

Resumo

O presente trabalho especifica as etapas do processo de montagem de uma bicicleta com três diferentes arranjos – convencional, elétrica e ergométrica –, além da elaboração de um sistema de monitoramento e controle em Android, avaliados como os principais componentes do projeto. Este trabalho tem por finalidade apresentar uma sugestão ecológica e sustentável aos problemas de mobilidade urbana.

O diferencial do protótipo de bicicleta apresentado neste trabalho são os modos estacionário e elétrico, onde no estacionário se acumula a energia originada pelas pedaladas e a transforma em eletricidade através de um gerador acoplado à roda traseira, possibilitando o uso do estilo elétrico, o qual visa ajuda na locomoção do ciclista.

É interessante pensar em algum tipo de supervisório a fim de verificar o rendimento do sistema e acompanhar o desenvolvimento do ciclista. Isso é feito na forma de um aplicativo para Android, o qual também realiza a comunicação entre o usuário e a máquina no controle do acionamento do motor e da velocidade do mesmo. Para tanto, utilizam-se sensores e circuitos sob o comando de um microcontrolador ATmega328P. O sistema embarcado irá se conectar a um *smartphone* via *bluetooth*, sendo capaz de enviar e receber dados.

Palavras-chave: Bicicleta elétrica, Android, Arduino, controle, monitoramento.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A palavra bicicleta, como enunciado em dicionários etimológicos, deriva do grego κυκλσ ("kyklos", círculo ou roda), através do latim tardio *cyclos*, e o prefixo latino "bi-"(dois), formando a correspondente inglesa *bicycle*, ou seja duas rodas; Neste contexto, uma bicicleta convencional pode ser definida como um veículo de duas rodas movido pelo esforço do usuário (ciclista) através do acionamento de pedais, uma elétrica (do inglês, *electricbike* ou $e-bike$), por outro lado, apesar da mesma constituição básica, recorre ao auxílio de um motor elétrico na sua locomoção. Tem sido destinadas aos mais variados usos ao longo do tempo, por exemplo, transporte utilitário (cargas e pessoas), "*workhorse*"(entregas gerais, paramédicos e polícia) e recreacional (turismo, esportes e competições), e seus aspectos técnicos dividem-se em tipo, dinâmica e performance, porém o objetivo principal da sua utilização tende a ser a rápida e eficiente locomoção.

Não se sabe ao certo quando ou quem a inventou, sendo citados Leonardo Da Vinci e o Conde Sirvac, contudo a primeira bicicleta dirigível remete ao Barão Karl Drais von Sauerbronn em 1816, um apetrecho movido por impulsão direta contra o solo; Em 1853, Philip Moritz Ficher acoplou aros metálicos à ambas as rodas e um par de manivelas à roda dianteira, transformando assim a bicicleta (PEQUINI, 2005). Constata-se, ao longo do tempo, uma melhora significativa na segurança, aerodinâmica, resistência e leveza do modelo, fora o avanço nas ferramentas associadas, como freios e marchas. A primeira $e-bike$ patenteada retrocede à 1895, por Ogden Bolton Jr., contudo o estopim de seu desenvolvimento se deu na década de 90 graças ao avanço tecnológico na área dos controladores, sensores e baterias. Quanto à aceitação no mercado, a Holanda, em termos percentuais, é a maior utilizadora de bicicletas no mundo, todavia, a China conta com um enorme número de usuários de bicicletas elétricas (ARAúJO, 2012).

A cerca das bicicletas convencionais, estima-se que o mercado nacional conta com um grande número de unidades, entretanto, as vendas e produções nacionais vem diminuindo consideravelmente nos últimos anos; Sabe-se também que o percentual de bicicletas do tipo brinquedo é o maior, porém boa parte do restante é utilizada para transporte (ABRACICLO, 2015). Quanto às *e-bikes*, no Brasil, o mercado ainda está em desenvolvimento e a maioria dos modelos presentes é importado, os kits de adaptação também estão cada vez mais populares. O Código de Trânsito Brasileiro (CONTRAN, 1997) trata, em vários momentos, a respeito de bicicletas e seus usuários, falando a cerca das prioridades e infrações, normas específicas ao seu uso, além dos equipamentos obrigatórios (campainha,

sinalização e espelho retrovisor). A Resolução nº 465 de 27 de Novembro de 2013 (DENATRAN, 2013) estabelece as normas de tráfego de veículos ciclo-elétricos.

De tal forma, é interessante destacar as vantagens de sua utilização tanto para o meio ambiente quanto para o bem-estar do usuário. Cita-se especialmente, quanto ao seu baixo impacto ao ecossistema, a redução da poluição do ar (a emissão de gases é da fonte fornecedora de energia elétrica), da água (sem risco de vazamento de fluidos nocivos), sonora (insignificantes ruídos e buzinas) e do desgaste das estradas (leve e pouco agressiva), em outras palavras, trata-se de um modo suave de transporte. Economizar dinheiro, escapar de congestionamentos e buscas por vagas de estacionamento também são algumas vantagens a serem mencionadas; A mobilidade urbana é otimizada, uma vez que um carro ocupa o espaço de várias bicicletas. É classificada então como um meio de transporte urbano sustentável (SILVA, 2013). Aponta-se a prática do ciclismo como vinculada à numerosos benefícios ao corpo, dentre eles a minimização do estresse sobre o coração, aumento do HDL no sangue e diminuição dos triglicéridos, diminuição da gordura corporal, redução da ansiedade e depressão, melhora na sensação de bem-estar e, além disso, melhora na produtividade no trabalho, diversão e atividade física (ULL, 2015).

No quesito da locomoção em si, a bicicleta é vista como uma das mais eficientes máquinas de conversão de esforço humano em propulsão, estima-se que apenas 1% da energia transmitida à roda traseira é perdida, o que torna fácil manter uma velocidade cerca de quatro vezes maior do que a do caminhar (PEQUINI, 2005). Evidencia-se, entretanto, o cansaço ao utilizador em distâncias mais longas e, neste intuito, tem-se a bicicletas elétrica, a qual permite que qualquer pessoa (independentemente de faixa etária e preparo físico) percorra um espaço maior à custos de transporte menores comparados à outros veículos. Denota-se também, a partir da mesma afirmação, que é completamente viável a produção e armazenamento de energia elétrica por meio da energia mecânica proveniente das pedaladas, apesar de que atualmente a tecnologia dos geradores não é tão eficiente para este fim específico de montagem (uma quantidade de energia é perdida na forma de calor, por exemplo) e ainda possui um preço consideravelmente alto.

A ideia de monitoramento surge com a necessidade de avaliar o rendimento do conjunto, $e-bike$ e ciclista (KIEFER; BEHRENDT, 2015). No quesito da máquina, espera-se a obtenção de uma eficiência ótima nas circunstâncias de trajeto, rota e velocidade exigidas pelo usuário, tornando favorável a sua utilização como meio de transporte sob as outras opções disponíveis. Quanto no acompanhamento do desenvolvimento do indivíduo é visualizado o estímulo a hábitos saudáveis, analisa-se então fatores como distância e intervalo de tempo de pedalada, o que simplifica casos em que é preciso assistência e orientação. É rentável também a observação de características ambientais, complementando as informações com as condições às quais o exercício foi submetido. Dá-se então o uso de aplicativos específicos, enriquecendo assim a quantidade e qualidade de conteúdo ao qual o ciclista tem acesso e garantindo a interação homem-máquina de forma personalizada.

Neste intuito, apresenta-se aqui o projeto de uma bicicleta elétrica e um aplicativo de monitoramento para Android que atenda às ambições do ciclista e que seja elaborada com teor similar aos padrões estabelecidos pela legislação brasileira. Estima-se também a confecção de um aparato auxiliar para se ligar a bicicleta em um modo estacionário destinado à geração de energia, de maneira a carregar baterias.

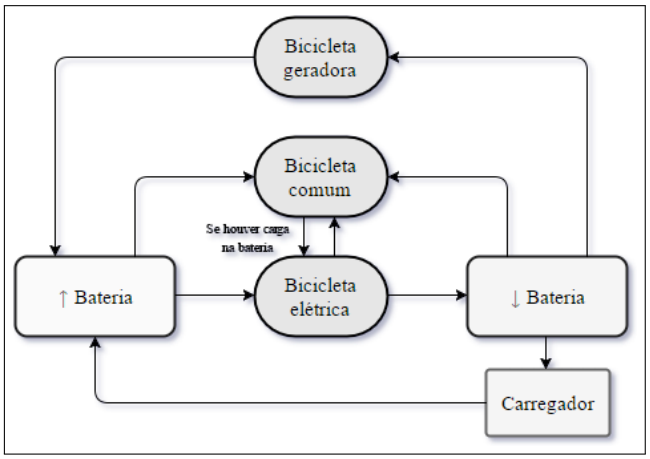
Capítulo 2

Revisão Bibliográfica

2.1 Análise física e funcional do modelo

Sabendo que um dos objetivos deste projeto é a elaboração de uma bicicleta com 3 modos, é importante o levantamento de informações sobre a constituição básica e funcionalidade de cada um, o que visa o entendimento da montagem e atuação dos circuitos de controle e aquisição de dados. Destaca-se que o modo elétrico só pode ser utilizado caso a bateria esteja cheia, o modo gerador caso ela possua pouca carga e o modo comum em qualquer situação (Figura 2.1).

Figura 2.1 – Visão geral da configuração dos modos da bicicleta.



2.1.1 Bicicleta convencional

Como já explicitado, trata do modelo básico, situação onde o veículo se move apenas pelo esforço do usuário. A bicicleta adotada para o ensaio é a Fischer Hill Razer (Figura 2.2), cujas especificações são apresentadas na Tabela 2.1.

Figura 2.2 – Modelo da bicicleta Fischer Hill Razer Dual Suspension Aro 26.

Tabela 2.1 – Informações técnicas da bicicleta Fischer Hill Razer Dual Suspension.

Aro	Alumínio, 26”
Quadro	Aço carbono, 16”
Marcha	21 velocidades
Garfo	Com suspensão
Guidão	MTB em aço carbono
Freios	V-Brake
Manoplas	PVC flexível
Corrente	1/2x3/32 IMP
Selim	MTB
Pedivela	170mm
Engrenagem	Tripla
Pneus	26x2.125
Raios	2mm
Rodas	Alumínio
Pedal	Rosca, 9/16”, plástico com refletor
Movimento central	34.7mm, eixo de ponta quadrada
Dimensões aproximadas	104x60x17.2cm
Peso líq. aproximado	16.8kg

Aponta-se também que sua estrutura física básica conta principalmente com os seguintes componentes:

- Quadro: Tubo designado à receber os principais componentes da bicicleta;
- Selim: Assento;
- Mesa: Vínculo entre o guidão e o garfo;
- Guidão: Tubo fixo à parte superior do garfo, direciona os movimentos da bicicleta;
- Garfo: Posto da roda;
- Roda: Transmite a força aplicada no eixo para a borda, sendo composta por pneu, aro, raios e cubo;
- Cubo: Elemento central da roda, eixo;
- Pedal: Acomodação para os pés do ciclista, conectado ao eixo do movimento central;
- Passador de marchas: Mecanismo de mudança das engrenagens da catraca;
- Catraca: Conjunto de anéis dentados;
- Coroa: Aro dentado que transmite o movimento da pedalada para a corrente;
- Corrente: Grupo de elos metálicos e flexíveis que conecta a catraca à roda traseira;
- Câmbio: Sistema que direciona a corrente na coroa ou no pinhão;
- Freio e manetes de freio: Trava de segurança;
- Amortecedor: Mola de absorção de impactos.

2.1.2 Bicicleta elétrica

Também já retratada, a *e-bike* remete à utilização de um motor elétrico na locomoção ou auxílio. A bicicleta montada se encaixa no tipo "*moped*", uma vez que o usuário tem as opções de, exclusivamente, pedalar ou acelerar. Nesta lógica, a análise dos componentes ativos (ARAúJO, 2012) é um fator decisivo na definição de variáveis de projeto, são eles:

- Motor elétrico: Máquina com o propósito de converter energia elétrica em mecânica;
- Controlador de motor: Elemento com a função de energizar as fases do motor no momento indicado, a fim de fazer o rotor do motor dar uma volta completa;
- Sensor de posição: Indicativo da posição instantânea do rotor do motor;
- Baterias: Dispositivos que armazenam energia elétrica através das reações eletroquímicas que se processam no seu interior, fornecendo-a quando necessário;
- Sistema de pedal assistido (PAS): Acionamento do motor a partir da pedalada;
- Painel de instrumento: Sistema de acompanhamento em tempo real das informações da bicicleta.

O motor em questão funciona baseado nos princípios do magnetismo, assim, ao ser percorrido por uma corrente elétrica, o fio condutor (neste caso, bobinas) imerso em um capo magnético, se move pela ação de uma força perpendicular a ele. O motor mais utilizado para bicicletas elétricas é o Corrente Contínua Sem Escovas, popularmente chamado de BLDC (*Brushless Direct Current*), modelo que não possui escovas e cujo rotor é feito de ímãs permanentes; O qual é síncrono (o campo magnético gerado pelo estator

e formado pelo rotor giram na mesma frequência) e necessita de um circuito de controle eletrônico (seus enrolamentos devem ser energizados seguindo uma sequência), e se torna vantajoso pelo fato de não gerar ruído e faíscas, não possuir um limitador da velocidade máxima do motor ou do número de polos existentes no estator. Optou-se por um motor próprio para bicicletas (já acoplado ao aro dianteiro), cujas especificações se encontram na Tabela 2.2.

Tabela 2.2 – Especificações do motor

Potência	250W
Tensão nominal	36V
Dimensões	15x9.5x15cm
Massa	3.7kg

Os controladores de motores voltados para tal aplicação são dispositivos que controlam o funcionamento do motor em função de determinadas variáveis, as quais dependem fundamentalmente da topologia do motor e/ou de imposições legais. No quesito topologia do motor, os controladores podem tratar simplesmente de um botão de ligar/desligar ou controlar todos os parâmetros do motor de forma autônoma. O módulo controlador em questão, descrito na Tabela 2.3, é próprio para o motor usado.

Tabela 2.3 – Especificações do controlador do motor.

Tensão	36V
Carga	15Ah
Defasagem do sinal	120^{o}

Quanto às baterias, dividem-se em duas vertentes, primárias (fornecem energia elétrica uma única vez) e secundárias (toleram um número finito de recargas), e, quanto aos elementos químicos que as compõem, fala-se das baterias de chumbo, lítio, níquel-cádmio e níquel hidreto-metálico. Sua característica elétrica mais relevante a capacidade de armazenamento de energia, definida através da corrente que a mesma pode fornecer durante um período de tempo (Ah) e pela tensão que a mesma dispõe aos seus terminais (V). Para o projeto utilizaram-se baterias seladas de chumbo-ácido gel e de ciclo profundo com 12V e 10Ah ligadas em série para atingir os 36V requeridos pelo motor.

O painel instalado consiste em um conjunto *smartphone* e suporte, dado que se elaborou um aplicativo para Android que trata das informações julgadas necessárias ou interessantes. Um Arduino recolhe e processa os dados obtidos com o uso de sensores, e os encaminha, via *bluetooth*, para o aplicativo.

2.1.3 Bicicleta geradora

Propõe-se um sistema de geração que não agride o meio ambiente e ao mesmo tempo proporciona uma atividade física para o usuário (Figura 3.6). Partindo da premissa de que é praticável a geração de energia por meio de uma bicicleta estacionária (BARROS;

SILVA, 2016), supõe-se um cenário onde um ou mais usuários (membros de uma família, por exemplo) poderiam se exercitar em uma bicicleta ergométrica acoplada à um gerador e fornecer energia para a residência, aparelhos eletrônicos ou para carregar baterias. A bicicleta utilizada para a geração de energia é a mesma do restante do projeto, sua roda traseira é colocada em contato com um suporte com os componentes necessários para o armazenamento da energia.

- Gerador: Dispositivo utilizado para a conversão da energia mecânica, química ou de outras formas em elétrica;
- Conversor de tensão: Circuito eletrônico que converte uma tensão (AC ou DC) que tem uma determinada amplitude, em outra tensão do mesmo tipo com outra amplitude;
- Retificador: Elemento que permite que uma tensão ou corrente alternada seja transformada em contínua;
- Baterias.

Fez-se uso de um aerogerador, em outros termos, um Gerador de Magnéticos permanente Neodym (*neodymium magnets*), cujos detalhes estão dispostos na Tabela 2.4.

Tabela 2.4 – Especificações do gerador ISTABREEZE i500.

Modelo	i-500G
Conexão	3 fases
Potência de saída	400W
Potência máxima	500W
Tensão	12V AC
Amperagem máxima	41.7A DC
Temperatura de atuação	-40^{o}C a 60^{o}C
Peso	2kg

O controlador de carga (Tabela 2.5) é próprio para o gerador utilizado, trata de um conversor e regulador que controla a saída à 12V AC com 3 fios, e mantem a voltagem ideal que vai pra bateria. Outro detalhe é que, quando a bateria chega na carga certa, é enviado um sinal para o gerador que o "trava", impedindo de carregar mais, o que protege a bateria.

Tabela 2.5 – Especificações do controlador INSTABREEZE i500.

Tensão de decisão para sistema de 12V a 24V	17V
Tensão para início e final de carga de bateria	12.6V e 14.8V
Tensão mínima de detecção de bateria presente	8.5V
Tensão máxima de entrada	25V
Temperatura de desligamento por temperatura alta e baixa	80^{o}C e -30^{o}C
Temperatura de religamento por temperatura alta e baixa	70^{o}C e -20^{o}C
Compensação térmica de limites de carga da bateria	-0.033V/^{o}C
Faixas de compensação térmica	5 ^{o}C, 15^{o}C, 25^{o}C, 35^{o}C e 45^{o}C
Rotação mínima de turbina para indicar carga	300rpm
Rotação máxima de turbina	3000rpm
Potência máxima na entrada de turbina	500W
Tempo de frenagem proporcional	20s
Tempo de recuperação após condições de erro cessarem	2min

As baterias utilizadas são as mesmas do modo elétrico, contudo foram ligadas em paralelo para manter os 12V.

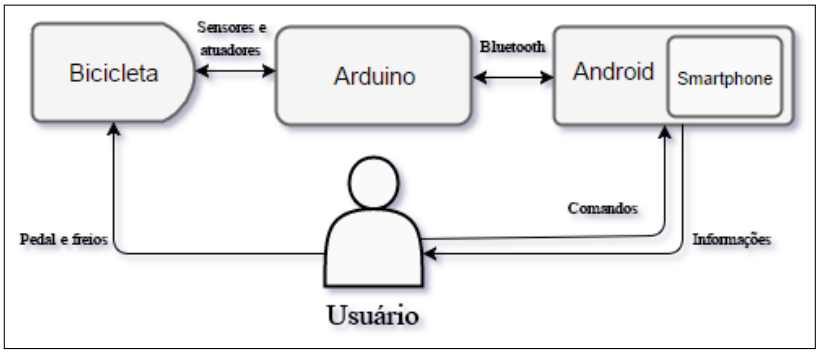
2.2 Sistema de supervisão e aquisição de dados

Sabendo que a área da automação explana os níveis de campo (instrumentos a controlar e de detecção), controle (elementos que controlam o processo) e supervisão (programas de interface homem-máquina e aquisição de dados), é dito que um sistema automatizado é aquele capaz de verificar seu próprio funcionamento sem a necessidade de um operador. Ainda neste raciocínio, diz-se que um sistema embarcado é um agrupamento microprocessado no qual o computador se encontra completamente dedicado ao conjunto controlado e realizando dadas tarefas com determinações intrínsecas, é inevitável a reprogramação de toda a estrutura quando se deseja alterar sua funcionalidade. De tal forma, pode-se reconhecer o conjunto proposto neste trabalho (Figura 2.3), o qual realiza medições e as salva em um banco de dados, apresenta quadros para comparação e análise, além de permitir o acionamento e controle das partes, tudo em uma interface simples e interativa.

2.2.1 Módulos, sensores e circuitos

Compõem a fundação de todo sistema de automatização, são dispositivos sensíveis à alguma forma de energia do ambiente capazes de associar dados à grandezas mensuráveis. São classificados entre analógicos e digitais, os primeiros assumem como sinal de saída, ao decorrer do tempo, qualquer valor dentro de sua faixa de operação, enquanto os outros admitem apenas "0"ou "1"(respectivamente, baixo e alto) como resposta. Menciona-se também que características como sensibilidade (ganho ou razão entre a saída e a entrada), exatidão (proximidade do valor obtido do verdadeiro), precisão (grau de repetitividade

Figura 2.3 – Visão geral do sistema.



na resposta) e linearidade (curva do instrumento em relação ao padrão) devem ser igualmente levadas em consideração. Citam-se aqui os sensores utilizados, bem como algumas especificações e fotos para referência.

- Módulo *bluetooth*: Comunicação entre o Arduino e o Android;
- Sensor de efeito Hall: Contagem de rotações por minuto, ou seja, no cálculo da distância e da velocidade da bicicleta;
- Sensor de temperatura: Indicação da temperatura ambiente;
- Sensor de luminosidade: Verificar a luminosidade do ambiente e enviar um aviso ao ciclista para ligar a lanterna da bicicleta caso o mesmo esteja baixo (<50% da faixa de atuação do sensor);
- Sensor de Monóxido de Carbono: Verificar o nível de CO no ambiente e acionar um alarme quando o mesmo estiver acima do limite estabelecido (100ppm), uma vez que esse gás é prejudicial à saúde;
- Circuito escada: Controlar de velocidade do motor (4 modos);
- Divisor de tensão: Verificar a carga da bateria.

HC-05

Adotou-se a comunicação *bluetooth* para ligar o Arduino e o *smartphone*. Uma transmissão sem fio por meio de radiofrequência, a qual possibilita a detecção entre dispositivos habilitados que estejam próximos independente de sua posição. Tecnicamente é um padrão de baixo consumo energético e baixa velocidade de transmissão de dados, sendo encontrado na maioria dos celulares modernos e o cujo módulo próprio para Arduino, neste caso o HC-05 (Figura 2.4), é de fácil utilização (Tabela 2.6) e relativamente barato.

Figura 2.4 – Módulo bluetooth HC-05.

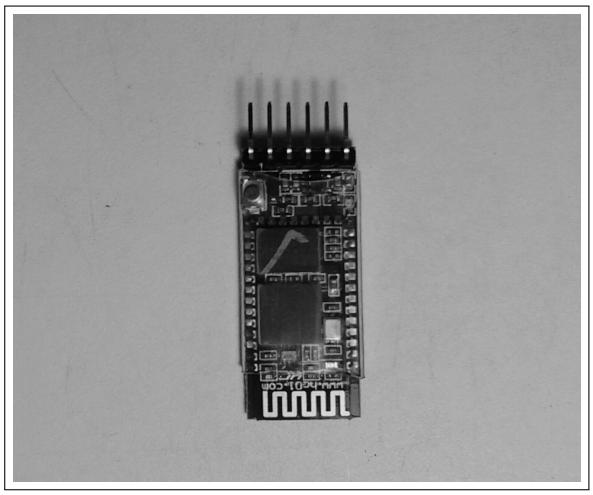


Tabela 2.6 – Especificações do HC-05.

Protocolo bluetooth	v2.0+EDR
Firmware	Linvor 1.8
Frequência	2.4GHz Banda ISM
Modulação	GFSK
Emissão de energia	<=4dBm, Classe 2
Sensibilidade	<=84dBm com 0.1%BER
Velocidade assíncrono	2.1Mbps(máx)/160kbps
Velocidade síncrono	1Mbps/1Mbps
Segurança	Autentificação e encriptação
Perfil	Porta serial bluetooth
Modo	Escravo e mestre
Banda de onda	2.4Hhz-2.8Ghz, Banda ISM
Tensão	3.3V (2.7-4.2V)
Corrente	Pareado 35mA, Conectado 8mA
Temperatura	-40 à +150^{o}C
Alcance	10m
Baud rate	4800;9600;19200;38400;57600;115200; 230400;460800;921600;1382400

Um detalhe é que antes de comunicar com o módulo é preciso pareá-lo com o dispositivo que se deseja conectar. Apesar disto variar dependendo do sistema operacional usado, em termos gerais é necessário:

1. Habilitar o *bluetooth* do dispositivo.
2. Procurar por outros dispositivos *bluetooth*.
3. Procurar por um dispositivo chamado "HC-05"e parear com ele.
4. Digitar o código "1234"

KY-035

A aplicação do sensor de efeito Hall no projeto se da no cálculo da distância percorrida e da velocidade, tanto do modo convencional quanto do elétrico. Trata-se de um transdutor cuja tensão de saída varia em resposta à aplicação de um campo magnético, atuando segundo a premissa do efeito Hall, isto é, a particularidade de um material que, na presença de um campo magnético perpendicular ao fluxo de corrente, expõe o desvio da trajetória das cargas. O modelo em questão é o KY-035 (Figura 2.5), todavia, por não ser um sensor oficial, não se encontram muitas informações à seu respeito (Tabela 2.7).

Figura 2.5 – Sensor de efeito Hall KY-035.

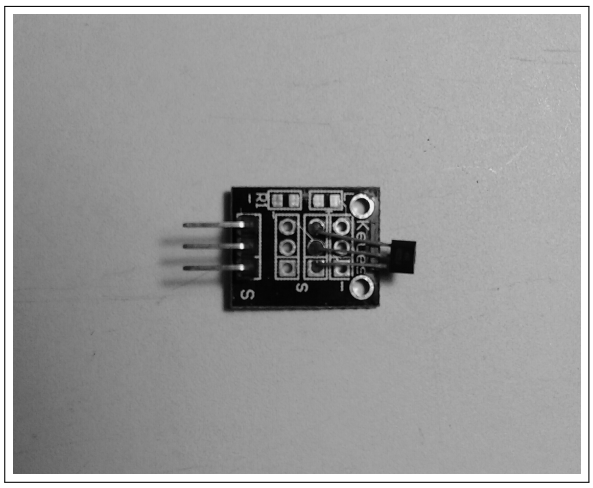


Tabela 2.7 – Especificações do KY-035.

Série	3144
Marca	KEYES
Classe	Bihor magnetic sensor
Tensão de operação	4.5-24V DC

LM35

Sensor com a tensão de saída linearmente proporcional à temperatura (Figura 2.6), utilizado para averiguar a situação do ambiente. De ligação simples e boa precisão, atua dentro de uma faixa aplicável à região de teste (Tabela 2.8).

Figura 2.6 – Sensor de temperatura LM35.

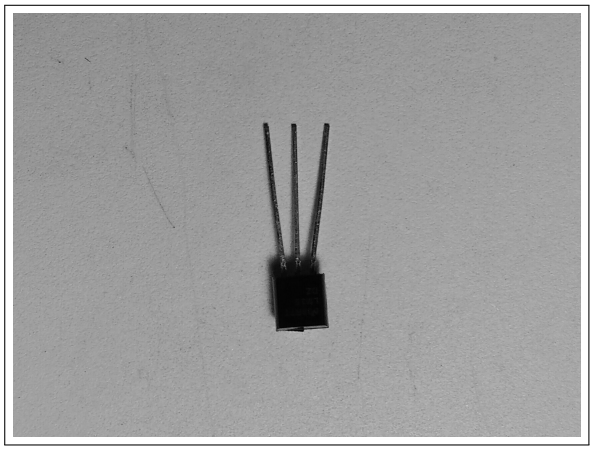


Tabela 2.8 – Especificações do LM35.

Faixa de temperatura	0-100^{o}C
Precisão	0.5^{o}C
Unidade	Graus Celsius
Tensão de operação	4-30V
Consumo de corrente	<60μA
Dimensões	4.3x4.3mm

LDR

Resistor dependente de luz (do inglês, *lightdependentresistor*) que age sobre o principio da fotocondutividade, em outras palavras, cuja resistência varia de acordo com a intensidade luminosa que o atinge, tendo sua curva de resposta espectral semelhante à do olho humano. As forças atuantes são inversamente proporcionais, assim, na escuridão total tem-se um valor máximo de resistência, já na presença luz intensa a resistência é mínima. O modelo GL5528 (Figura 2.7) é amplamente empregado devido ao seu baixo custo e facilidade de utilização, apesar de não ser tão preciso uma vez que sua medição pode variar devido à influência da temperatura e necessitar de um tempo de latência entre mudanças de iluminação. Contudo, seu uso é o suficiente para esta atividade, já que se supõe que o monitoramento será realizado em trajetos ao ar livre onde não são recorrentes as mudanças bruscas de iluminação e temperatura (Tabela 2.9).

Figura 2.7 – Sensor de luz LDR.

Tabela 2.9 – Especificações do LDR.

Diâmetro	5mm
Tensão máxima	150V DC
Potência máxima	100mW
Temperatura de operação	-30∼+70^{o}C
Consumo de corrente	60mA
Espectro	540nm
Resistência no escuro	1MΩ (lux 0)
Resistência na luz	10-20kΩ (lux 10)

MQ-7

Sensor de gás extremamente sensível a monóxido de carbono (CO), um composto altamente tóxico para o ser humano e que é um dos elementos produzidos na queima de combustíveis fósseis (gasolina, por exemplo). A utilização deste sensor visa o bem-estar do usuário enquanto pedala pela cidade, o qual poderá mudar de rota caso uma concentração muito alta seja captada. O modelo escolhido é o MQ-7 (Figura 2.8), detalhado na Tabela 2.10.

Figura 2.8 – Sensor de Monóxido de Carbono MQ-7

Tabela 2.10 – Especificações do MQ-7.

Comparador	LM393
Concentração da detecção	10-10000ppm
Tensão de operação	3-5V
Resistência de aquecimento	31Ω +/- 0.2Ω
Tensão de aquecimento	5V +/- 0.2V
Potência de aquecimento	$\leq$350mW
Sensibilidade	Ajustável (potenciômetro)
Saída	Analógica e digital
Resistência na luz	10-20kΩ (lux 10)
Dimensões	32x20x15mm

R2R

Uma *resistor ladder* é um circuito elétrico feito a partir de unidades de repetição de resistências em uma configuração tipo escada, é uma maneira simples e barata de realizar

a conversão digital-analógico. Tal método foi implementado para o controle da velocidade do motor, sendo utilizados resistores de 10kΩ.

Divisor de tensão

Trata de uma técnica de projeto utilizada para criar uma tensão elétrica (Vout) que seja proporcional à outra tensão (Vin). No projeto, este circuito foi utilizado para verificar a carga das baterias, uma vez que elas estão em um arranjo correspondente a 36V e o Arduino trabalha com 5V. Aplicaram-se dois resistores, um de 10kΩ e outro de 1kΩ, suportando assim uma tensão de até 50V.

2.2.2 Software e hardware

Este tópico trata do conjunto de programas usados no decorrer do projeto, destacando-se aqui os de maior contribuição.

Arduino

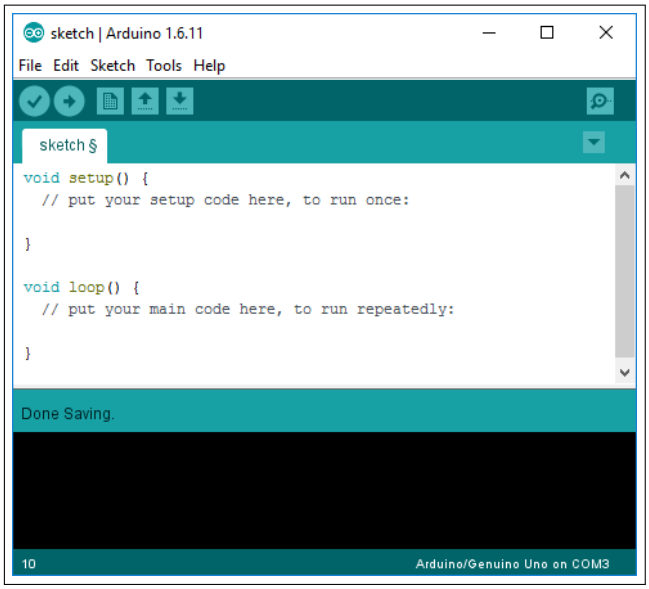
O Arduino é uma plataforma eletrônica acessível que opera em diferentes sistemas operacionais (Windows, Macintosh OSX e Linux), com ambiente de programação explícito e compreensível (simples, porém versátil), além de *software* e *hardware* de código aberto e extensível. As placas são gerenciadas ao se enviar instruções para o microcontrolador através do Arduino Integrated Development Environment - ou Arduino Software (IDE) - utilizando-se a linguagem de programação Arduino (ARDUINO, 2016).

Na IDE (Figura 2.9), há 5 menus com comandos (*File*, *Edit*, *Sketch*, *Tools* e *Help*), além de uma barra principal da IDE que conta com as seguintes funções:

- Verify: Confere o código para erros de compilação.
- Upload: Compila e carrega o código para a placa.
- New: Cria um novo rascunho.
- Open: Abre a janela de projetos para a escolha de um, a fim de acioná-lo.
- Save: Salva o rascunho atual.
- Serial monitor: Exibe uma tela com dados serial que estão sendo enviados.

O *File* trata dos projetos no geral, nele se encontram as opções de abrir um novo editor (*new*), abrir um projeto (*open* e *open recent*), abrir a pasta de projetos (*sketchbook*), bem como as alternativas salvar (*save* e *save as*) e fechar (*close* e *quit*). O *Edit* refere-se especificamente ao rascunho em edição, pode-se desfazer e refazer ações recentes (*undo/redo*), colocar ou remover o marcador de comentário na seleção (*comment/uncomment*) adicionar ou subtrair um espaço no início das linhas selecionadas (*increase/decrease indent*), além de procurar e substituir algum item (*find*). Na *Sketch* é possível verificar, compilar e carregar o código para a placa (*verify/compile* e *upload*), bem como incluir bibliotecas (*include library*). Em *Tools* são realizadas as formatações (*auto format*), seleção da placa (*board*) e seleção do dispositivo serial (*port*). E, por fim, a *Help* conta com a seleção da página na cópia da referência para a função ou comando sob o cursor (*find reference*).

Figura 2.9 – Janela inicial da IDE do Arduino.



As especificações da *board* do Arduino UNO (Figura 2.10) são apresentadas na Tabela 2.11.

Figura 2.10 – Board do Arduino UNO.

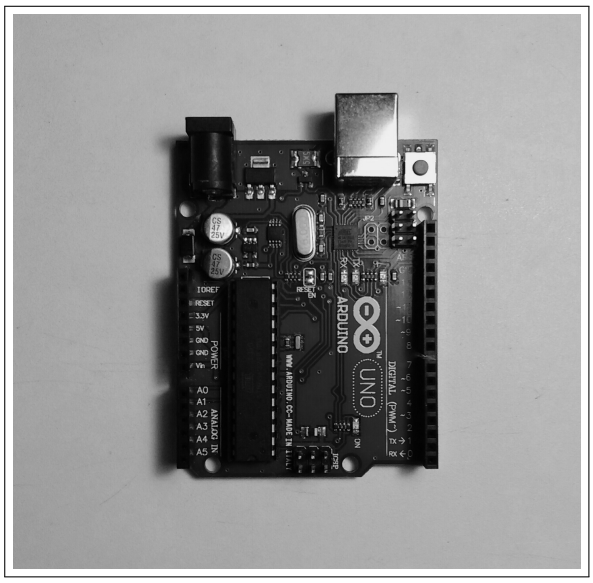


Tabela 2.11 – Especificações do Arduino UNO.

Microcontrolador	ATmega328
Tensão de operação	5V
Tensão de entrada (recomendada)	7-12V
Tensão de entrada (limit)	6-20V
Pinos digitais I/O	14
Pinos digitais PWM I/O	6
Pinos de entrada analógica	6
Corrente DC por pino I/O	40 mA
Corrente DC para pino 3.3V	50 mA
Flash Memory	32 KB
Flash Memory para Bootloader	0.5 KB
SRAM	2 KB
EEPROM	1 KB
Velocidade do clock	16 MHz

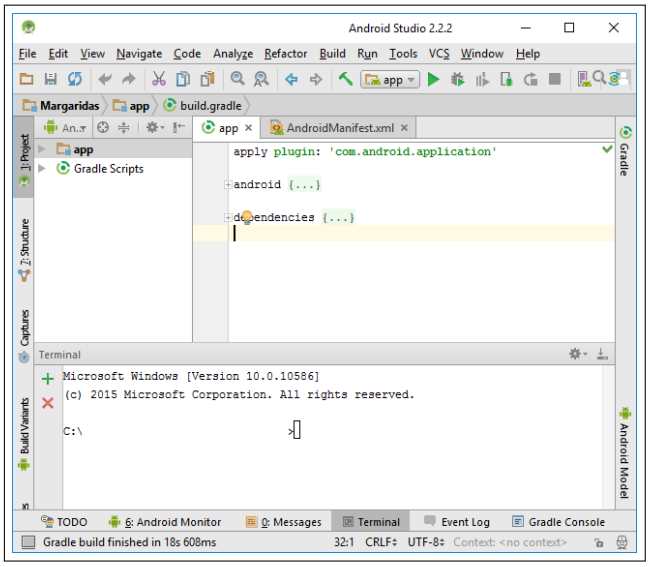
Android Studio

Plataforma oficial de desenvolvimento Android, ou seja, a IDE (Figura 2.11). As abas dispostas na interface são: *File*, *Edit*, *View*, *Navigate*, *Code*, *Analyze*, *Refactor*, *Build*, *Run*, *Tools*, *VCS*, *Window* e *Help*. A barra principal conta com funções como:

- *Open*: Abrir um projeto.
- *Save All*: Salvar o progresso.
- *Synchronize*: Sincronizar.
- *Undo* e *Redo*: Desfazer e refazer.
- *Make project*: Fazer projeto.
- *Run*: Enviar o projeto para o dispositivo.
- *Debug 'app'*: Depurar o aplicativo.

Na lateral esquerda se encontra a árvore do projeto, onde é possível acessar os *manifests*, o java (*activities*, principalmente), *assets* (como as *fonts*) e *res* (*drawable*, *layout* e *values*, por exemplo).

Figura 2.11 – Janela inicial do Android Studio.

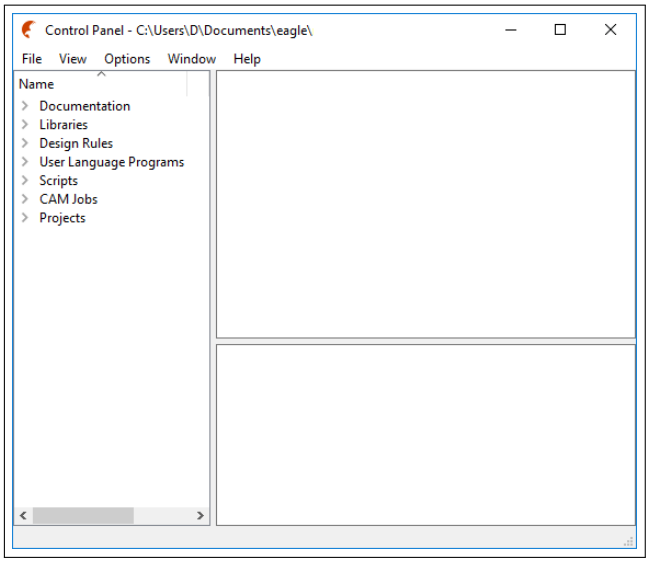


Seus recursos mais valorizados são: *gradle* baseado em suporte de compilação; refatoração específica do Android e correções rápidas; ferramentas para detectar desempenho, usabilidade, compatibilidade de versão e outros problemas; integração ProGuard e recursos de assinatura de aplicativos; modelos para criar designs e componentes comuns; editor de layout rico que permite aos usuários arrastar e soltar componentes de interface, opção para visualizar *layouts* em várias configurações de tela; suporte para criar aplicativos Android Wear; suporte integrado ao Google Cloud Platform, permitindo a integração com o Google Cloud Messaging e o App Engine.

Eagle

Na produção do *schematic* e do *layout* da placa adotou-se o software EAGLE (Easy Applicable Graphical Layout Editor), cuja janela inicial pode ser vista na Figura 2.12, uma ferramenta de design de PCB aberto e bastante flexível. Ele trabalha com apenas um layer e o tamanho da board já é predefinido.

Figura 2.12 – Janela inicial do EAGLE



No *schematic* é realizado o desenho do modelo, nele são especificados todos os componentes e ligações. Estima-se a fácil visualização do circuito para análise da sua funcionalidade, verificação de erros e substituição de componentes. Alguns dos atributos disponíveis nesta aba são o *save* (salvar), *gerate/switch to board* (gerar ou mudar para a *board*) e lupas de zoom. Contudo, as principais funções para a construção do modelo estão posicionadas na lateral esquerda da área de desenho, algumas delas estão descritas abaixo:

- *Add*: Adicionar componentes. Faz-se uma pesquisa na *library* (biblioteca) e seleciona-se o elemento mais próximo do ideal.
- *Move*: Posicionar as peças no local correto; pode-se também movimentar grupos.
- *Copy*: Copia o componente selecionado.
- *Rotate*: Rotaciona a peça em 90^{o} para a esquerda.
- *Delete*: Exclui as elementos selecionados.
- *Wire*: No modo net, ligam-se os componentes.
- *Name* e *value*: Os nomes e valores dos componentes podem ser alterados individualmente, basta apenas clicar neles.

Pode-se também alterar as configurações da rede no botão *grid*, como tamanho, unidade (milímetros ou polegadas) e estilo (pontos ou linhas). Recomenda-se a adoção de uma única unidade para todo o projeto, a fim de evitar erros tanto na prototipagem quando montagem do circuito.

Em seguida, promove-se o design da *board* a partir do *schematic*, que consiste na organização, posicionamento e conexão através de trilhas dos componentes. Aqui se busca a minimização do espaço tanto na disposição dos objetos quando no tamanho das trilhas e ilhas, uma placa pequena significa praticidade e funcionalidade. Esta aba também tra-

balha com funções, muitas delas iguais às disponíveis no *schematic*. Abaixo são citadas as que diferem e que mais foram utilizadas:

- *Route*: Via manual;
- *Ripup*: Encaminha trilhas;
- *Ratsnest*: Calcula o menor caminho para as trilhas;
- *Autorouter*: Rota automática, projeta as solucões mais viáveis para a conectividade dos elementos;
- *DRC*: Verificação de regras de design, definição dos tamanhos e distâncias entre as trilhas e ilhas.

Capítulo 3

Metodologia

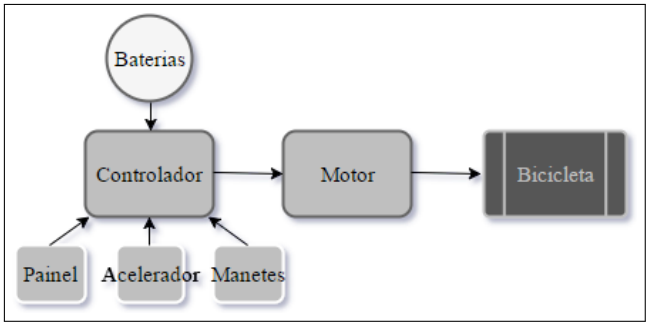
3.1 Montagem da bicicleta

A Figura 2.1 apresenta o sistema como um todo, porém é interessante observar detalhadamente cada parte que compõe o modelo de bicicleta proposto no projeto.

3.1.1 Modo elétrico

Na parte elétrica (Figura 3.1), todo o sistema é abastecido pelo conjunto de baterias. O controlador está ligado ao motor, o qual é acionado pelo *smartphone*. Detalha-se que os freios, quando ativados, desligam o motor.

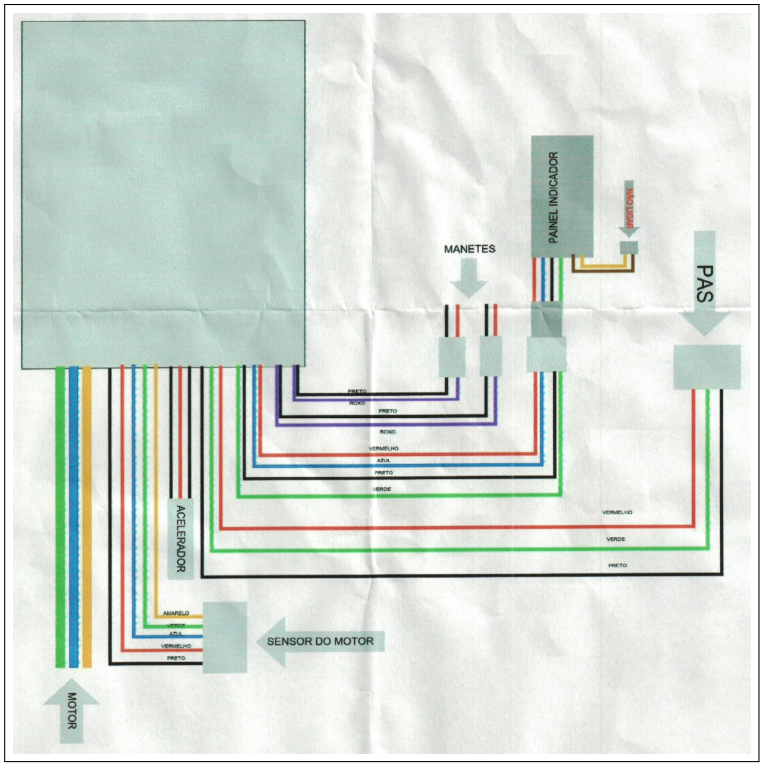
Figura 3.1 – Visão geral da configuração do modo elétrico da bicicleta.



Nas Figuras 3.2 e 3.3, podem ser vistos o protótipo e as ligações do controlador do motor.

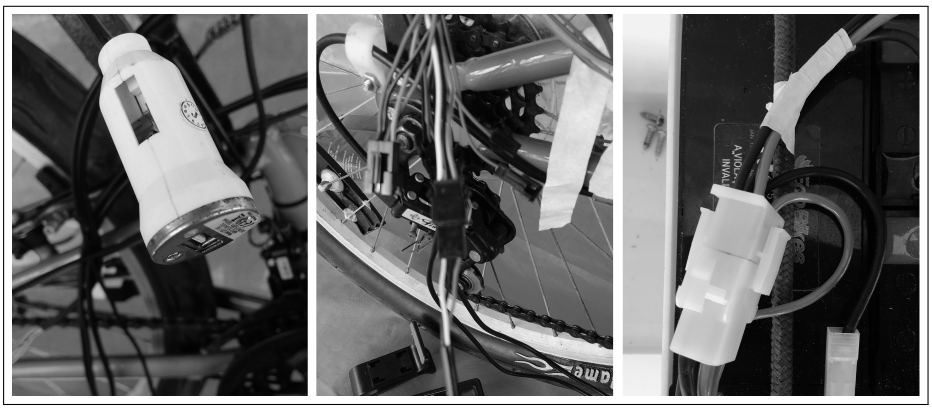
Figura 3.2 – Visão do protótipo da bicicleta elétrica.

Figura 3.3 – Esquema das ligações do controlador do motor.



A Figura 3.4 apresenta a alimentação do Arduino (conversor de 12V para 5V em formato USB, ligado à uma das baterias), os cabos utilizados para acionar o motor pelo Arduino (os quais partem para o acelerador e para o monitor, uma intervenção que consiste na retirada do acelerador manual) e, por fim, o *plug* para conexão das baterias (elaborado para facilitar a troca de modos) que foram ligadas em série.

Figura 3.4 – Visão das ligações da bicicleta elétrica.



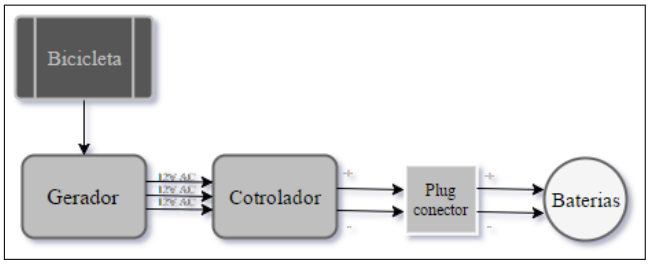
Na Figura 3.5 tem-se uma caixa branca com o sistema embarcado (com 3 saídas externas para o LDR, o LM35 e o MQ-7) e uma caixa preta com o controlador do motor.

Figura 3.5 – Visão das caixas com o sistema embarcado e o controlador do motor.

3.1.2 Modo gerador

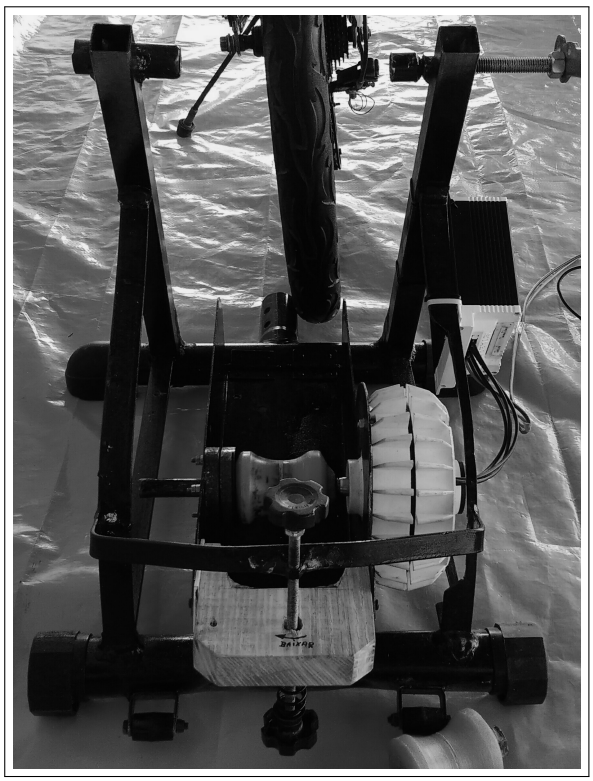
Quanto ao modo ergométrico (Figura 3.6), a roda traseira da bicicleta se encaixa em um conjunto gerador de 12V DC, composto pelo gerador e controlador.

Figura 3.6 – Visão geral da configuração do modo gerador da bicicleta.



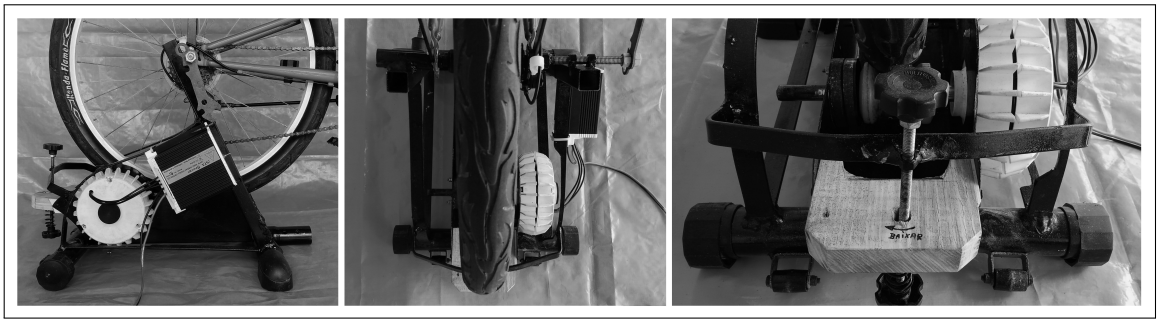
O protótipo pode ser visto na Figura 3.7, e conta com dois sistemas de ajustes da posição da bicicleta, sendo um na lateral de fixação na base e outro na parte posterior.

Figura 3.7 – Visão do suporte gerador sem a bicicleta.



Em detalhes (Figura 3.8), realiza-se o contado do eixo gerador com o pneu traseiro da bicicleta, de forma que cada giro do pneu proporciona 13 voltas no eixo. Tal eixo remete ao gerador ligado à um rodete, o qual foi feito artesanalmente usando duas rodas de *skate* com 62mm de diâmetro e 37mm de espessura, uma barra de ferro de 18mm de espessura, uma porca sextavada de 28mm e massa durepoxi.

Figura 3.8 – Visão do suporte gerador com a bicicleta acoplada.



3.2 Aquisição de dados no Arduino

Toda a parte de sensoreamento foi feita utilizando Arduino, por tanto, destacam-se neste tópico os pontos importantes referentes à montagem do circuito e da programação.

3.2.1 Conexões

Quanto às conexões dos dispositivos na *board* do Arduino, os pinos digitais utilizados foram os de 2 a 10, respectivamente:

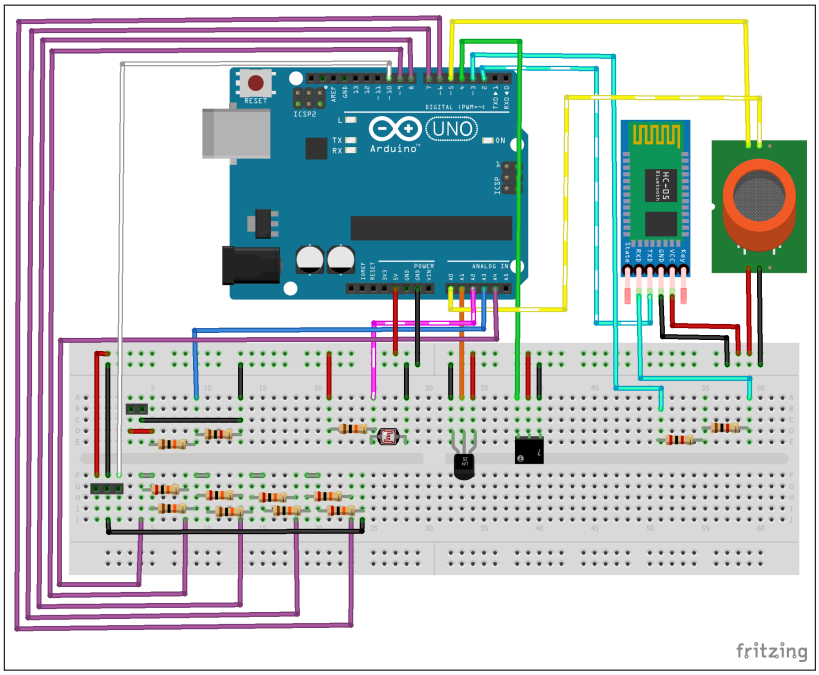
- RXD do HC-05
- TXD do HC-05
- SIGNAL do KY-035
- DOUT do MQ-7
- Saída digital 1 do R2R
- Saída digital 2 do R2R
- Saída digital 3 do R2R
- Saída digital 4 do R2R
- V0 do controlador do motor que iria para o acelerador

E os digitais de 0 a 5:

- AOUT do MQ-7
- V0 do LM35
- Saída do LDR
- Saída do divisor de tensão
- Saída V0 do R2R

A Figura 3.9 representa o esquema de ligação, nota-se que o sistema também conta com um GND e um Vcc comum.

Figura 3.9 – Esquema geral do sistema embarcado.



3.2.2 Inicialização

Na IDE, inicialmente declararam-se as bibliotecas, constantes, variáveis e o cabeçalho das funções, cujas finalidades podem ser presumidas a partir dos nomes. Quanto às bibliotecas, a `#include <Wire.h>` permite a comunicação com dispositivos I2C/TWI, enquanto a `#include<SoftwareSerial.h>` possibilita a comunicação serial em outros pinos digitais do Arduino usando software. As constantes correspondem aos pinos e valores que não serão alterados ao longo do processo, ou seja, valor dos resistores para o cálculo da voltagem no divisor de tensão. As variáveis são as *flags* e os valores obtidos a partir da leitura dos sensores (medidos ou calculados). Criaram-se também sete *headings* para as funções de análise de dados.

3.2.3 Setup

Função de configuração, aqui se inicia a comunicação serial e se definem os pinos como *input* ou *output*.

3.2.4 Loop

Trata da função principal, é a responsável por chamar as outras funções repetidamente e iniciar a comunicação serial e a *bluetooth*.

3.2.5 Android Comunication

Acessa-se o módulo *bluetooth*, e, enquanto ele estiver ativado, lê uma *string* "c"contendo os dados enviados pelo Android. Cada mensagem corresponde à uma ação:

- *on*: Ligar o motor.
- +1: Aumentar a velocidade em 1.
- -1: Diminuir a velocidade em 1.
- *off*: Desligar o motor.

Também é enviada uma *string data* contendo os dados das outras funções.

3.2.6 Hall Sensor

Esta função apenas conta o número de vezes que o sensor KY-035 foi acionado, ou seja, as voltas completas da roda. Tal valor é enviado para o Android, que calcula a distância percorrida até o momento e a velocidade medida.

3.2.7 Carbon Monoxide Sensor

Trata da leitura dos valores obtidos no sensor MQ-7, os quais já se encontram nas unidades correspondentes. Utilizou-se o número informado pelo pino analógico, dado em ppm, na definição de nível alto ou baixo de gás no ambiente.

3.2.8 Temperature Sensor

Remete à tensão lida no LM35, a qual corresponde à temperatura, além de realizar a conversão para graus Celsius.

$$envTemperature = \frac{\frac{lm35Voltage}{1024}5000}{10} \tag{3.1}$$

3.2.9 Light Sensor

A luminosidade é calculada a partir do valor lido no LDR e dada em porcentagem.

$$envBrightness = (1 - \frac{ldrValue + 1}{1023})100 \tag{3.2}$$

Esse valor é enviado na forma de nível alto ou baixo.

3.2.10 Motor Control

A partir da Tabela 3.1, criaram-se condições *if* para a ordem de acionamento dos pinos segundo a informação recebida no *bluetooth*, a de saída corresponde à uma tensão entre 0V e, aproximadamente, 5V. Verificou-se que as influências do mcP0, mcP1, mcP2 e mcP3 são de, respectivamente, 0.3V, 0.62V, 1.24V e 2.5V.

Tabela 3.1 – Tabela verdade do controle do acionamento do motor

mcP3	mcP2	mpP1	mcP0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

O Android é o responsável por controlar os modos, definidos de 0 a 4 de forma a dividir as combinações apresentadas na tabela, acionando-os em rampa, tanto para o aumento como diminuição da velocidade.

3.2.11 Battery Monitoring

A partir do valor lido no *batRawValue*, realiza-se a conversão para um valor real de voltagem (*batVoltage*) e, por meio desse valor, calcula-se a porcentagem de carga na bateria.

$$batVoltage = \frac{batRawValue\frac{5}{1024}}{\frac{batR2}{batR1+batR2}} \tag{3.3}$$

$$batPercentage = \frac{batVoltage}{36}100 \tag{3.4}$$

3.3 Manufatura da placa de circuito impresso

Explicitam-se aqui as etapas de produção da PCB. Como citado, utilizou-se o EAGLE para o design da PCB e o método de transmissão térmica para a confecção.

3.3.1 Testes em protoboard

Após adquirir os componentes eletrônicos que compõem o sistema em estudo, é necessário testar se estão todos funcionando e averiguar se o *layout* da placa é implementá-

vel fisicamente, para tanto se monta o circuito em uma *protoboard* (Figuras 3.10 e 3.11), onde se podem alterar as posições dos componentes sem perdê-los.

Figura 3.10 – Protoboard com o sistema de sensoreamento e aquisição de dados.

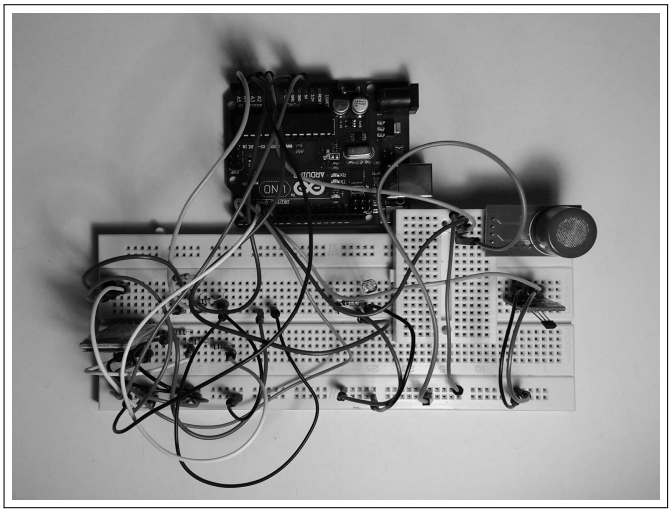
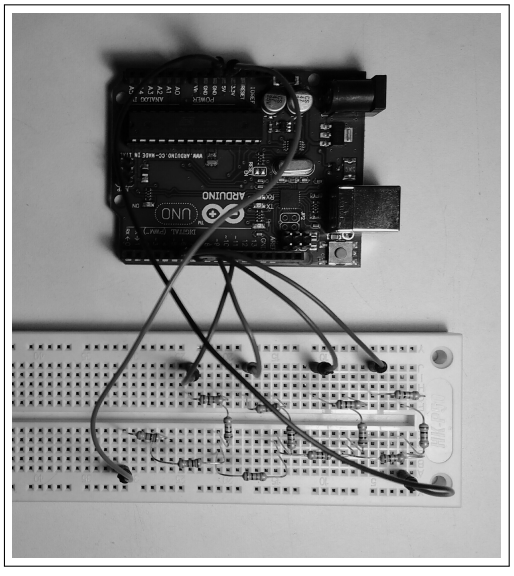


Figura 3.11 – Protoboard com o sistema de controle da velocidade.

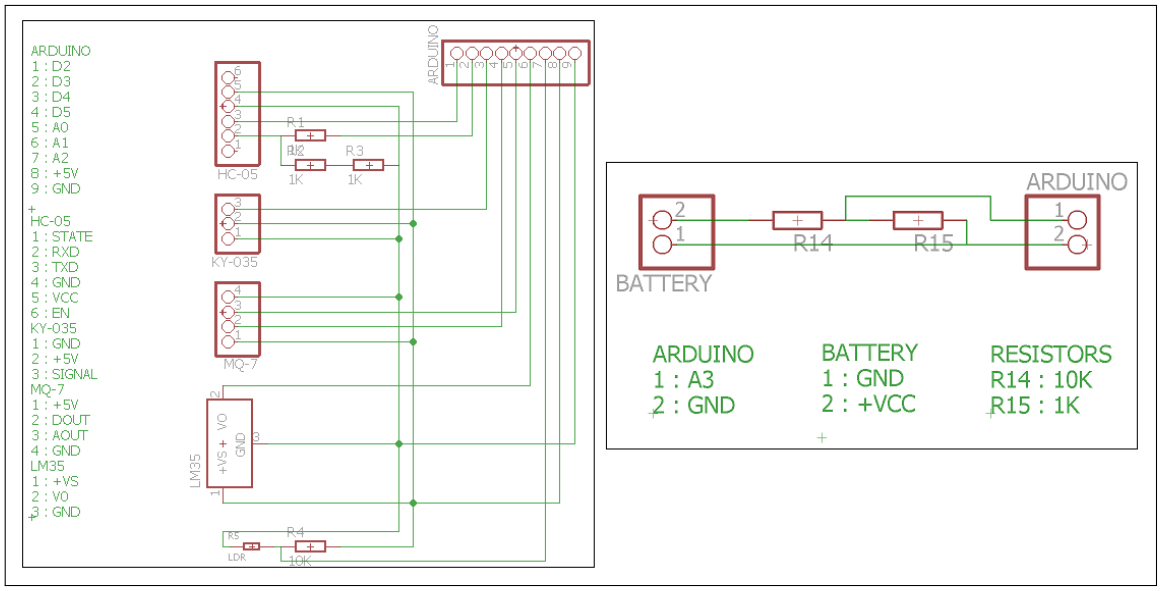


3.3.2 Schematic

Quanto ao *layout*, resumidamente, para a construção do *schematic* os componentes foram selecionados seguindo a ordem em que eles aparecem da esquerda para a direita,

copiou-se os que se repetem e, finalmente, eles foram posicionados e ligados. Os componentes que não foram encontrados na *library* foram substituídos por outros do mesmo tipo e encapsulamento. Desenharam-se duas placas (Figuras 3.12) visando uma melhor acomodação na caixa de suporte da bicicleta.

Figura 3.12 – Schematics do sistema de monitoramento.



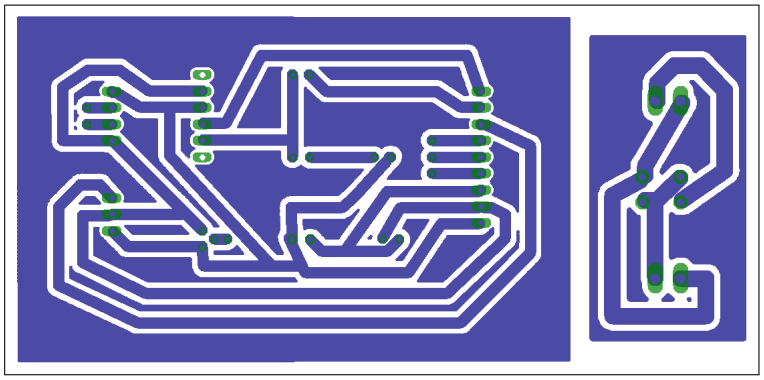
3.3.3 Board

Nas *boards* (Figura 3.13) do circuito em questão, tomou-se como orientação o *schematic* feito no passo anterior, pegou-se o componente que inicializa o circuito e posicionando-o aonde foi determinado o início da PCB, em seguida foi selecionado o componente que é ligado a ele, tal raciocínio foi executado durante toda a organização da PCB, escolhendo o próximo componente conectado (mostrado no *schematic*) e posicionando as pernas conectas em uma orientação que permitisse deixar os componentes próximos e de fácil conectividade.

Para o tamanho das trilhas e ilhas, foi utilizado neste projeto dimensões diferentes das que vem por definição no EAGLE. Tendo por experiência de projetos anteriores, notou-se a dificuldade de soldar componentes devido as ilhas serem pequenas e as trilhas muito finas, causado curtos. Como solução foram alteradas essas dimensões no *DRC*, na aba *Clearance* na opção *Wire* alterou-se o *Wire* para 25mil e em *Sizes* modificou-se a opção *Minumum Width* para 35mil (tamanho das trilhas). Diante disto, após finalizar a organização da PCB, a conexão por trilhas fica fácil devido ao comando *Autorouter*, contudo, as trilhas do *Top* foram feitas individualmente. Ainda percebendo que as trilhas não estão adequadas, define-se a *Width* da opção *Change* como 1.6764 e clica-se em cada trilha para

alterar sua espessura. E por fim, faz-se um isolamento com o auxílio da função *Polygon*, neste caso de 0.6mm.

Figura 3.13 – Layout das boards do sistema de monitoramento.



Como a fabricação das PCB se deu pelo método de transmissão térmica, o arquivo para impressão (que deve ser invertido ou espelhado) pôde ser gerado de duas formas: na aba *File*, seleciona-se a opção *Export* e o item *Image*, lá se dá um nome à imagem, escolhe-se a opção *Monochrome* e aumenta-se a resolução para 2400dpi; A outra solução consiste em optar pelo item *Print* também na aba *File*, criando assim um arquivo em pdf, neste caso se pode selecionar a localização da *board* no papel.

3.3.4 Confecção da placa

Com o *layout* final definido e os devidos testes realizados, parte-se para a confecção da placa. A impressão do *layout* é feita através de uma impressora a laser em um papel especial (normalmente, papel couchê ou fotográfico de gramatura 150), usa-se a qualidade máxima do toner a fim de evitar falhas. Impressoras de jato de tinta não podem desempenhar essa função.

Com a placa no tamanho ideal, deve-se retirar toda a gordura desta com uma esponja de aço e detergente (em água corrente) ou álcool; Não se pode tocar na face limpa novamente. O desenho é posicionado e fixado sobre o lado cobreado da placa e o conjunto é aquecido, com o auxílio de um ferro de passar roupas em temperatura média, até que toda a tinta seja transferida do papel para a placa, o que leva cerca de 15 a 30 minutos - aquecer a placa antes de colocar o papel diminui o tempo quando se passa o ferro -. Em seguida, leva-se o conjunto a um recipiente com água, esperando que a tinta solte do papel; E, por fim, retira-se o papel junto com qualquer resquício e, caso haja necessidade, pode-se retocar as falhas com caneta de tinta plástica (usualmente, caneta de retroprojetor).

Em um recipiente de plástico ou vidro, prepara-se uma solução de corrosivo (percloreto de ferro) em água nas proporções indicadas na embalagem, comumente 250ml de água para 100g de percloreto de ferro. Aquecer o percloreto de ferro acelera esse processo. A placa deve ficar imersa neste líquido por aproximadamente 15 ou 20 minutos, contudo esse tempo varia com a idade do corrosivo uma vez que este pode ser reutilizado,

tendo sua eficácia reduzida. Após o descanso, a placa deve ser lavada com água, sabão e a lã de aço suavemente.

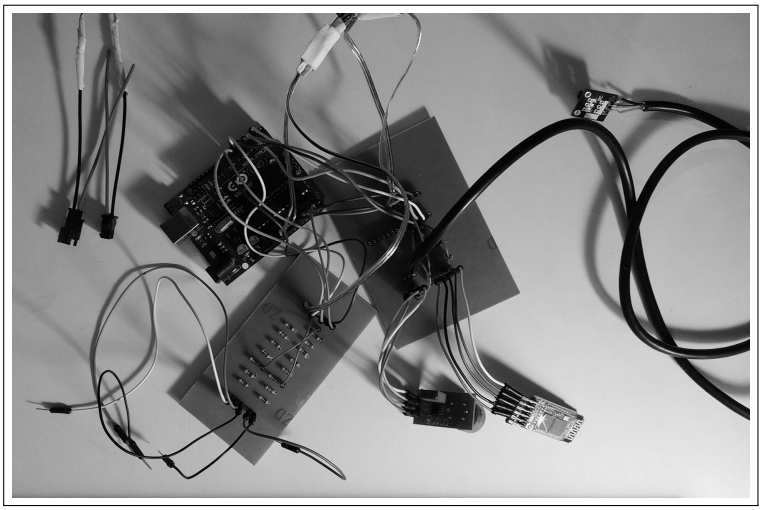
Os centros das ilhas e os furos demarcados nas bordas serão furados com um furador. Deve-se também lixar as bordas da placa.

Pode-se aplicar um produto antioxidante na superfície da placa, ou seja, estanhar a placa, para proteger o metal. Para tanto, se põe a pasta de solda na placa limpa e se espalha um pouco de solda nas partes com cobre. Caso essa etapa tenha sido esquecida antes da montagem do circuito, pode-se aplicar verniz ou esmalte do tipo base, tomando cuidado com as ligações.

A última etapa do projeto é a montagem do circuito na placa. Dispõem-se os componentes nos locais estabelecidos no desenho computacional da *board* e eles são soldados. Algumas dicas são: manter a ponta do ferro de solda no componente a ser soldado, mantendo a maior superfície de contato possível, e em seguida colocar o estanho; não usar muita solda, usar apenas o necessário; e limpar sempre o ferro de solda, o que vai melhorar a condução.

As PCBs prontas podem ser vistas na Figura 3.14.

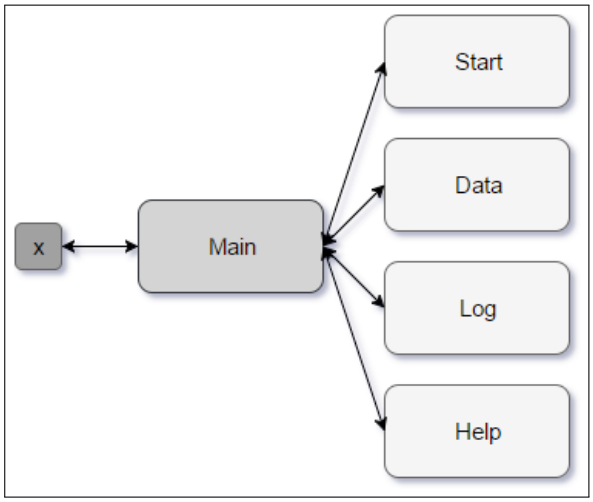
Figura 3.14 – Primeira PCB do sistema de monitoramento.



3.4 Elaboração do aplicativo

O aplicativo foi feito em Android (Figura 3.15), utilizando as linguagens Java e xml. Nesta seção são apresentados alguns detalhes da programação (GOOGLE, 2004).

Figura 3.15 – Activities do aplicativo.



3.4.1 Permissões

Para uma aplicação utilizar recursos e ferramentas do dispositivo (como câmera, *bluetooth* e GPS), primeiramente, ela deve ter a permissão para acessar os mesmos. Todas as permissões que o aplicativo requere são declaradas no arquivo `AndroidManifest.xml`, que também contém outras informações fundamentais, como o nome, ícone, *activity* inicial e tema. Para esta aplicação, foram utilizadas cinco permissões, que podem ser definidas em três grupos:

- *Bluetooth*: Para o uso de todas as funções do dispositivo *bluetooth* do celular, foram necessárias as permissões `BLUETOOTH` e `BLUETOOTH_ADMIN`, permitindo a localização, o pareamento e a troca de dados entre dispositivos.
- GPS: Habilita o uso do dispositivo GPS do celular, com o intuito de receber a localização do mesmo, em termos de latitude e longitude e, a partir delas, localizar o usuário no mapa. A permissão necessária é a `ACCESS_FINE_LOCATION`.
- Gerenciamento de arquivos: Para criar e escrever um arquivo na memória interna do celular ou em um dispositivo de memória externa (cartão SD, por exemplo), são necessárias as permissões `WRITE_EXTERNAL_STORAGE` e `WRITE_INTERNAL_STORAGE`. Elas devem ser utilizadas em conjunto, para fazer com que a aplicação possa, primeiramente, verificar se o arquivo existe, e, somente depois, escrevê-lo.

Após a permissão ser definida, o usuário deve aceitar que o aplicativo utilize o recurso, antes do mesmo ser usado. Isso geralmente acontece com uma *pop-up*, ou com o redirecionamento à tela de habilitação do recurso.

3.4.2 Banco de dados

Implementa-se o sistema de banco de dados a partir de uma classe criada dentro do aplicativo, a qual conta com métodos `static` para livre acesso de outras classes. O banco

de dados utilizado foi o SQLite, uma versão mais compacta do que a MySQL (geralmente utilizada em computadores).

O SQLite é instanciado como um objeto que rotula o nome do banco através de uma *string* e, no caso do Android, fica armazenado internamente no dispositivo com os arquivos ocultos. Para acessar o banco, o usuário deve utilizar a função `openOrCreateDatabase` e, para as operações de inserção de informações e criação de tabelas, aplica-se a função `execSQL` com uma *string* correspondente ao comando SQL desejado. Operações de seleção devem ser depuradas utilizando um objeto *Cursor*, que itera através de todos os resultados da busca.

3.4.3 Main Activity

Após a tela de abertura (*SplashScreen*), o usuário é levado a essa *activity*. Ela contém quatro botões, divididos de forma a ocuparem o espaço da tela igualmente, onde cada um redireciona o usuário a sua *activity* correspondente.

Outra característica importante da *MainActivity* é o fato dela nunca ser excluída da execução do aplicativo. Isso se deve às outras *activities* serem geradas a partir dela, fazendo com que a *main* continue salva internamente caso o usuário deseje voltar ao menu (apertar o botão de retorno do dispositivo). Essa característica não é vista nas outras *activities*, pois, ao sair das mesmas, o aplicativo exclui suas informações para poupar memória do dispositivo.

Tendo em vista a exploração dessa característica, muitos objetos, nos quais se há a necessidade de persistência da informação entre *activities*, são criados na *main activity* e acessados entre classes por meio da definição `public static`.

3.4.4 Start Activity

Responsável por gerar a localização do usuário através do sistema de GPS e mostrá-la em um mapa, juntamente com um marcador e uma linha que define a trajetória de locomoção. A mesma também é responsável por receber e enviar dados ao microcontrolador, utilizando uma comunicação *bluetooth*.

Um detalhe é que o usuário somente deve utilizar essa função caso o microcontrolador esteja ligado, com o risco do aplicativo parar de funcionar caso contrário. O aplicativo utiliza o nome do dispositivo a ser conectado (no caso, "HC-05"), o qual deve ser definido previamente.

Há também quatro botões, responsáveis por: ativar o motor, aumentar ou diminuir a velocidade do mesmo e interromper a corrida, os quais enviam uma *string*, que será decodificada pelo microcontrolador. Toda a comunicação com o dispositivo *bluetooth* é feita através de uma função, que é executada dentro de uma *thread* com duração de 1s, o aplicativo depura a *string* recebida, imprimindo-as nas *TextViews*, e envia comandos, caso o usuário tenha acionado algum botão (Figura 3.16).

Figura 3.16 – Pseudocódigo da função do bluetooth no Android.

```
   Entrada: String a ser enviada (S_out)
   Saída: TextViews (T_v)
 1 início
 2 |  Procurar e parear com o dispositivo
 3 |  para cada 1s de execução faça
 4 |  |  S_in = String recebida do dispositivo
 5 |  |  Transformar S_in em várias substrings
 6 |  |  T_v recebem suas substrings correspondentes
 7 |  |  se S_out não for nulo então
 8 |  |  |  Envia S_out para o dispositivo
 9 |  |  fim
10 |  fim
11 |  Encerrar conexão com o dispositivo
12 fim
```

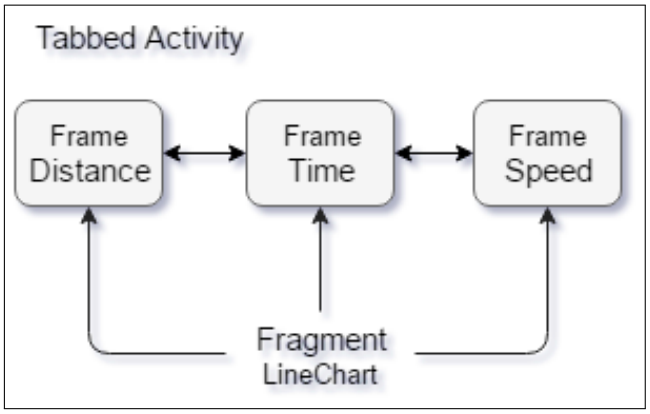
Através de uma fusão de APIs, há a união das API do Google Maps (define o mapa), com a API do sistema de localização, representada por um objeto `LocationListener`, utilizado para executar uma determinada função toda vez que o usuário mudar de posição. A utilização do serviço do Google Maps pode ser feita em tempo real, com o download do mapa durante a execução da aplicação, ou através do download prévio da localização, utilizando o aplicativo Maps, assim, dispensando o uso de internet.

O usuário tem total liberdade para sair dessa *activity*, já que as informações só serão apagadas da tela caso o mesmo acione o botão, além de armazená-las no banco de dados com essa ação.

3.4.5 Data Activity

Esta *activity* foi criada de forma a se ter um sistema de abas deslizantes, onde o usuário pode visualizar graficamente o seu rendimento ao longo da semana (Figura 3.17). Tais gráficos são gerados a partir da biblioteca aberta *MPAndroidChart*, que oferece uma variedade de gráficos, e depura a informação através de objetos próprios de entrada (*Entry*) e conjunto de dados (*DataSet*). O gráfico pode ser facilmente customizado, com métodos nativos, e pode ser carregado na aplicação através de um *fragment*.

Figura 3.17 – Formato da activity Data.



As informações geradas no gráfico são importadas diretamente do banco de dados, sendo que cada aba dessa *activity* é composta do mesmo *fragment*, porém populado com dados diferentes baseado na aba onde o usuário está observando no momento (Figura 3.18). O sistema de abas é criado automaticamente, na aplicação, com o uso de uma *tabbed activity*. Tal *activity* consiste em um layout que possui uma sensibilidade à ação de deslizar a tela para os lados, podendo executar determinadas funções baseados nessa ação.

Figura 3.18 – Pseudocódigo da DataActivity no Android.

```
   Entrada: Referência da tela observada R_i
   Saída: Gráfico (g)
1  início
2      Localizar o fragment e instancear os objetos
3      I = Lista de strings
4      se R_i for a primeira tela então
5          I = Informações do banco referentes a distância
6          Popular g com I
7      fim
8      se R_i for a segunda tela então
9          I = Informações do banco referentes ao tempo
10         Popular g com I
11     fim
12     se R_i for a terceira tela então
13         I = Informações do banco referentes a velocidade
14         Popular g com I
15     fim
16 fim
17 retorna g
```

3.4.6 Log Activity

Contém apenas um botão que, ao ser apertado, cria um arquivo `log.txt` na memória interna do dispositivo. Esse arquivo contém as informações de todos os percursos feitos pelo usuários e salvos no banco de dados, da semana atual. O documento é gerado na pasta inicial do armazenamento interno.

3.4.7 Help Activity

Composta somente de uma caixa de texto, com informações sobre aplicativo e instruções de como utilizá-lo.

Capítulo 4

Resultados

4.1 Bicicleta

Os principais testes de desempenho do modo elétrico da bicicleta se deram com o acionamento do motor e o uso do acelerador (Figura 4.1). Por meio destes, o funcionamento da *e-bike* foi garantido, averiguando-se que seu desenvolvimento é melhor quando há pouca carga sobre a bicicleta, a bateria está com carga total e a corrida é realizada em terrenos planos; Estima-se que a velocidade máxima obtida com o acelerador é de aproximadamente 25km/h, em condições ótimas.

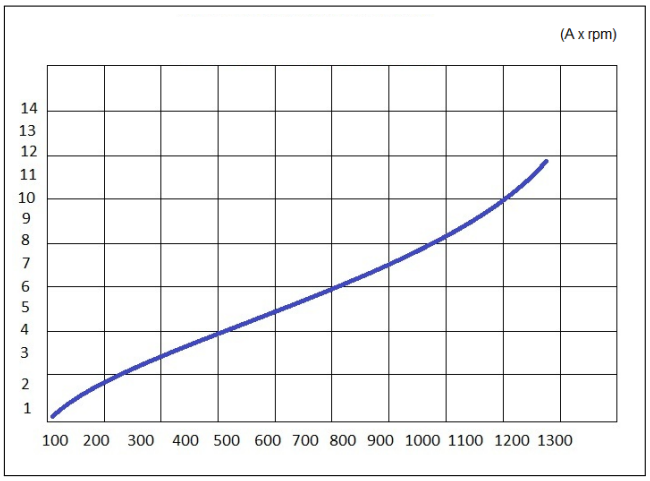
Figura 4.1 – Teste de funcionamento da bicicleta no modo elétrico com acionamento pelo acelerador.

Também foi realizado um experimento a cerca da comunicação com o Android, ligando o motor e aumentando o nível de velocidade nos devidos botões do aplicativo, a qual provou ser eficiente. Para tanto, suspendeu-se a roda dianteira da bicicleta, enquanto os botões eram pressionados em um curto intervalo de tempo (Figura 4.2).

Figura 4.2 – Teste de funcionamento da bicicleta no modo elétrico com acionamento pelo Android.

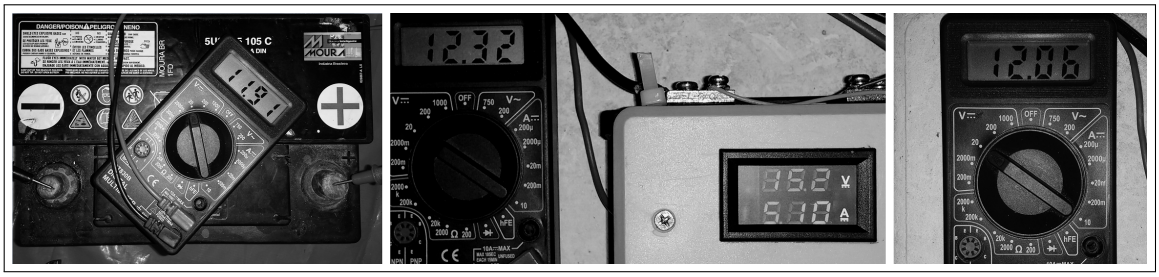
Quanto à parte de geração de energia, sabe-se que o gerador funciona segundo a relação apresentada na Figura 4.3.

Figura 4.3 – Relação de carga por RPM do gerador ISTABREEZE i500.



Um dos testes consistiu em carregar uma bateria de 12V, no qual, após 5 minutos de pedalada média, obteve-se o resultado exposto na Figura 4.4. Na primeira parte da imagem se tem a voltagem da bateria antes do processo, em seguida tem-se a carga sendo mandada para a mesma, e, por fim, a carga final da bateria.

Figura 4.4 – Teste de funcionamento da bicicleta no modo de geração de energia.

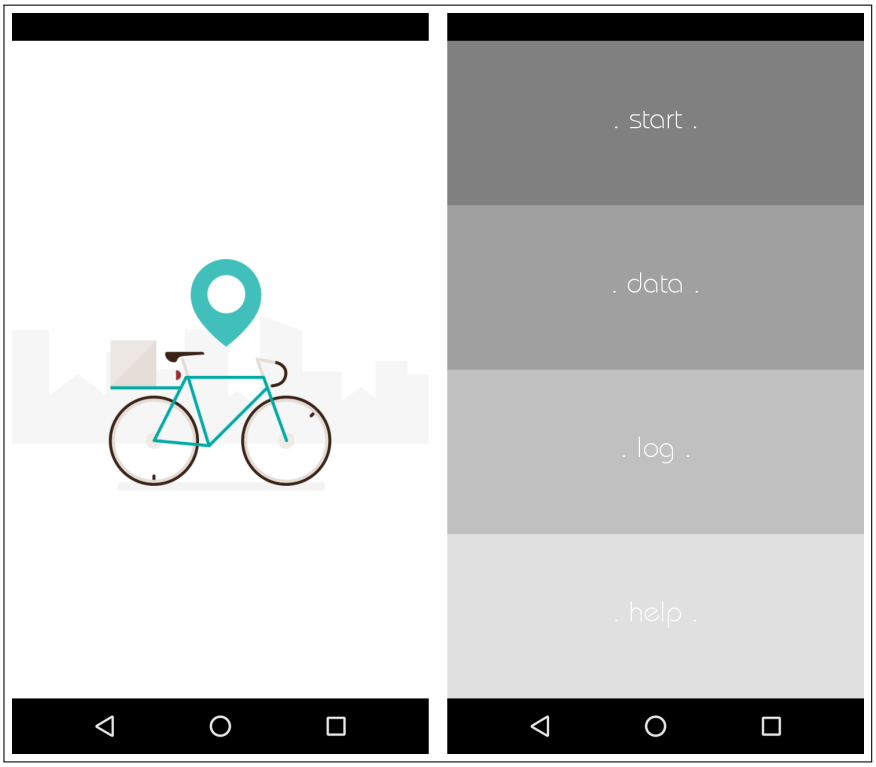


Através de experimentos desse gênero, verificou-se que a faixa de atuação, ou seja, o nível atingido através da pedalada, é entre 800 e 900rpm. O controlador direciona uma voltagem de cerca de 14V para as baterias, não ultrapassando muito esse valor caso a rotação esteja além do limite estabelecido. Também foi notado que, uma vez que a bateria esteja com menos de 8.5V, o controlador não é capaz de detecta-la. Como esperado, a geração de energia não foi tão eficiente (GIBSON, 2011), mas satisfez aos propósitos deste projeto.

4.2 Aplicativo

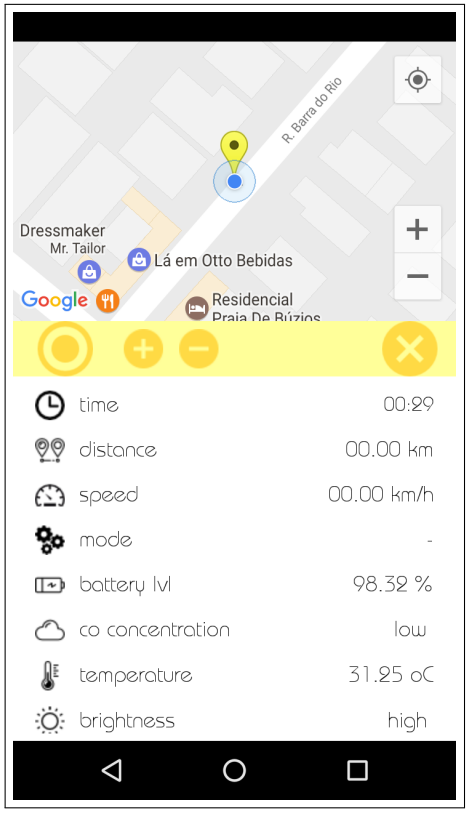
Ao abrir o aplicativo, inicialmente é apresentada a *SplashScreen*, seguida de um menu com quatro botões (MainActivity). Na esquerda da Figura 4.5 é mostrada a splash e à direita tem-se a main.

Figura 4.5 – Inicio do aplicativo.



Ao selecionar a opção *start*, o Android começa a receber as informações do Arduino, sendo atualizado a cada segundo. O cronômetro é iniciado e a temperatura ambiente, a porcentagem da carga da bateria, o nível da iluminação e o de CO são apresentados, também podem ser vistas a distância percorrida e a velocidade estimada.

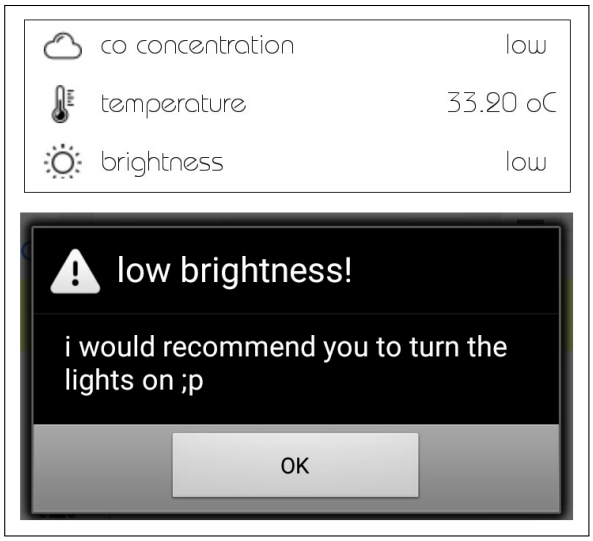
Figura 4.6 – Aba Start do aplicativo.



Caso o nível se iluminação esteja muito baixo, uma *pop-up* aparece na tela sugerindo que o ciclista ligue a lanterna da bicicleta. Utilizaram-se apenas duas configurações, alto ou *high* (>50%) e baixo ou *low*(<50%), e este aviso só aparece uma vez a cada troca de nível. O teste realizado para verificar o funcionamento do circuito consistiu na iluminação direta do sensor com o auxílio de uma lanterna (Figura 4.7), o que garantiu a mensagem de nível alto, seguido da remoção da fonte de luz (Figura 4.8), levando à indicação de nível baixo e aparição da *pop-up*.

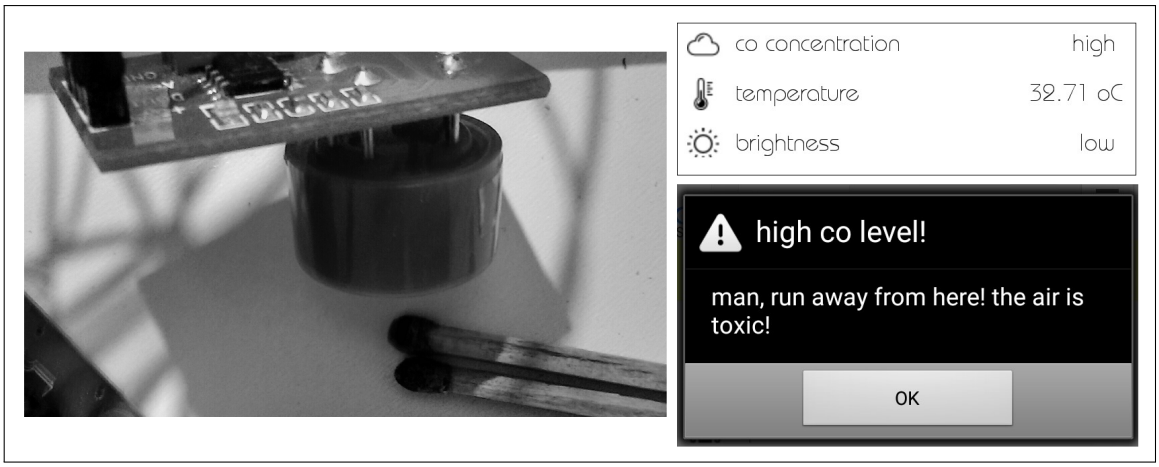
Figura 4.7 – Resposta da start do app para detecção de alta iluminação através do LDR.

Figura 4.8 – Resposta da start do app para detecção de baixa iluminação através do LDR.



Outra *pop-up* aparece na tela se for detectado um nível alto de CO no ambiente, a qual sugere que o usuário procure uma rota alternativa, uma vez que altos níveis de CO são tóxicos para o ser humano. Essa *pop-up* trabalha com o limite de 100ppm, e, como a citada anteriormente, aparece na tela uma vez a cada troca de nível . Para averiguar o desempenho do circuito, inicialmente se mediu a quantidade de CO no ambiente, que estava baixa, e, em seguida, foram ligados e apagados dois palitos de fósforo, os quais foram aproximados do sensor e levaram à indicação de nível alto e ao aparecimento da *pop-up* (Figura 4.9).

Figura 4.9 – Resposta da start do app para detecção de alto nível de CO através do MQ-7.

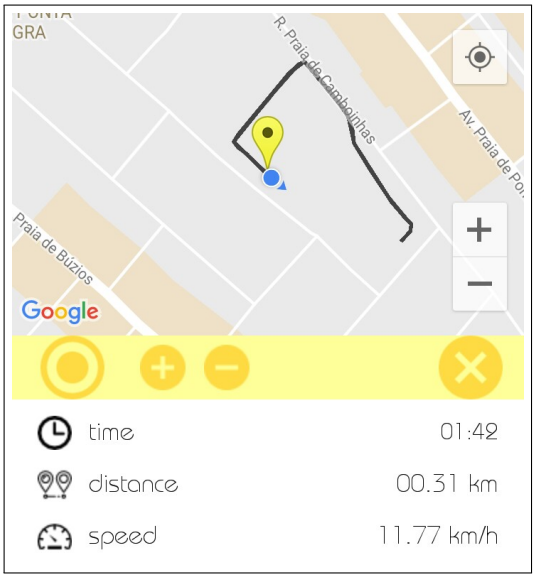


Mais duas *pop-up* aparecem quanto a carga da bateria está baixa (Figura 4.10). A primeira quando a voltagem de cada bateria está em 8.5V, informando que se o usuário quiser utilizar a base geradora para carregá-las, é preciso que ele desligue o motor. E a segunda quando a carga das baterias está por volta dos 30%, dizendo que é necessário desligar o modo elétrico para não danificá-las.

Figura 4.10 – Resposta da start do app para detecção de carga baixa na bateria.

Quanto ao desenho da rota, todo o trajeto é riscado no mapa quando é detectada a movimentação do ciclista, também há opções de aumentar ou diminuir o zoom, percorrer o mapa e focar no usuário (Figura 4.11).

Figura 4.11 – Resposta da start do app para movimentação do usuário.

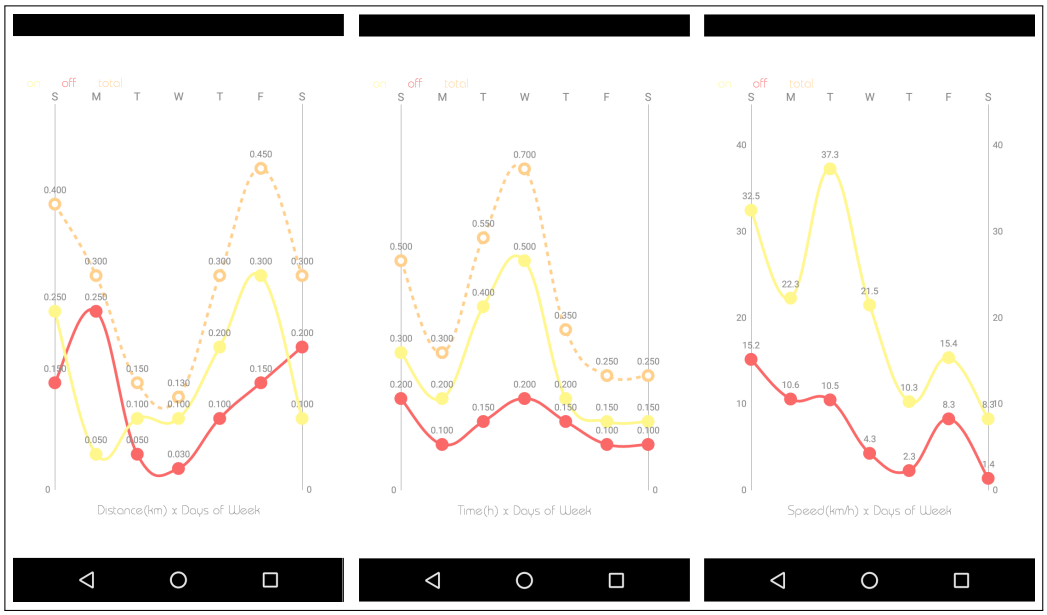


Na aba *data* tem-se os gráficos de desempenho, os quais inicialmente começam zerados (Figura 4.12).

Figura 4.12 – Abas do Data do aplicativo.

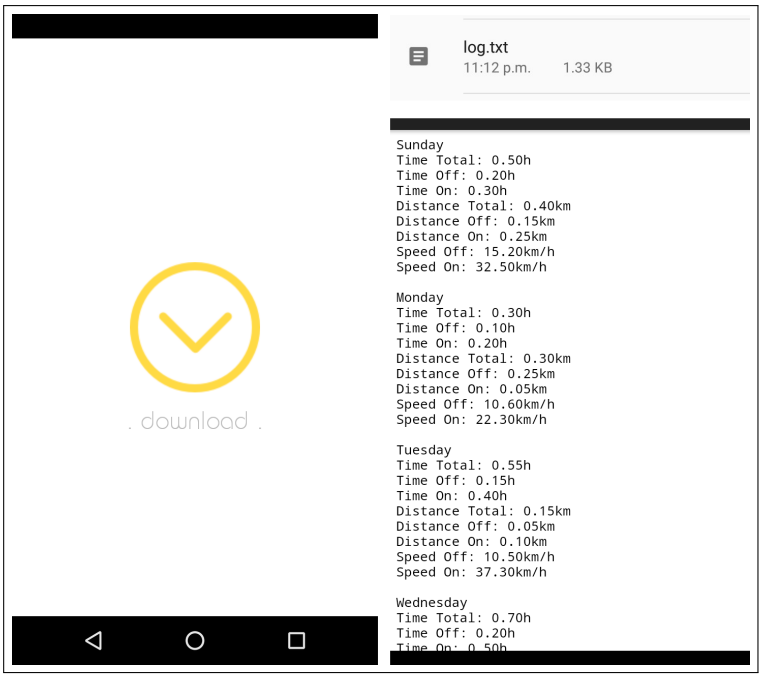
Para testar o funcionamento desta função, adicionaram-se valores aleatórios no banco de dados (Figura 4.13).

Figura 4.13 – Resposta da data do app para o banco de dados.



Os mesmos valores foram utilizados para o teste da *log*.

Figura 4.14 – Aba e resposta da Log do app para o banco de dados.



Capítulo 5

Conclusões

Através deste estudo pode-se demonstrar a importância de pesquisar meios de transporte alternativos, tanto por questões ambientais, de saúde e econômicas.

Os objetivos deste trabalho foram atingidos, uma vez que os três modos da bicicleta funcionaram e foi desenvolvido um sistema de monitoramento controlado pelo microcontrolador Atmega328P com interface em Android, o qual foi programado para mostrar a rota do ciclista, receber e apresentar os dados dos sensores, salvar informações em um banco de dados e, quando requisitado, salvá-las no celular.

Através da integração dos sistemas, notou-se a inviabilidade do acionamento do motor através Android por meio de botões, uma vez que tal condição seria desconfortável e até perigosa para o ciclista, já que seria necessário tocar a tela do *smartphone* para ligar a *e-bike* e aumentar ou diminuir a velocidade, ou seja, ele precisaria retirar uma das mãos do guidão e tirar os olhos da pista. Contudo, este modo é apenas para testes iniciais e o problema poderia ser resolvido com o acionamento por comando de voz, um objetivo futuro.

Constatou-se também que a base geradora é aplicável, porém, devido ao alinhamento do eixo do gerador com seu rodete que encosta no pneu, produziu um um ruído sonoro consideravelmente alto, necessitando de melhoramentos, a fim de elevar a quantidade de rpm, gerando mais energia e diminuindo o tempo de carregamento das baterias. O modo elétrico se desenvolveu bem, funcionando conforme projetado.

Referências Bibliográficas

ABRACICLO. *Associação Brasileira dos Fabricantes de Motocicletas, Ciclomotores, Motonetas, Bicicletas e Similares*. 2015. Disponível em <http://www.abraciclo.com.br/>. Acessado em novembro de 2016.

ARAúJO, R. J. F. *Desenvolvimento de uma Bicicleta Elétrica*. 2012. Universidade do Minho.

BARROS, W. M.; SILVA, F. S. F. Pequena geração de energia elétrica – bicicleta sustentável. *EITEC*, 2016.

CONTRAN. *Código de Trânsito Brasileiro*. 1997. Disponível em <https://www.planalto.gov.br/ccivil_03/LEIS/L9503Compilado.htm>. Acessado em novembro de 2016.

DENATRAN. *Resolução nº 465 do Conselho Nacional de Trânsito*. 2013. Disponível em <http://www.denatran.gov.br/download/Resolucoes/Resolucao4652013.pdf>. Acessado em novembro de 2016.

GIBSON, T. *Turning Sweat into Watts*. 2011. Disponível em <http://spectrum.ieee.org/>. Acessado em novembro de 2016.

GOOGLE. *Google Developers*. 2004. Disponível em <https://developers.google.com/>. Acessado em novembro de 2016.

KIEFER, C.; BEHRENDT, F. Smart e-bike monitoring system: real-time open source and open hardware gps assistance and sensor data for electrically-assisted bicycles. *IET - Intelligent Transport Systems*, 2015.

PEQUINI, S. M. *Ergonomia aplicada ao Design de produtos: Um estudo de caso sobre o Design de bicicletas*. 2005. PAU-USP.

SILVA, J. C. M. *Sistema de mobilidade elétrica de duas rodas de elevada eficiência e performance*. 2013. Instituto Politécnico de Viseu - ESTGV.

ULL. *Beneficios del uso de la bicicleta*. 2015. Universidad de La Laguna.

Apêndice A

Informações adicionais

A.1 Códigos-fonte do Arduino

A.1.1 Inicialização do código

```
#include<SoftwareSerial.h>
#include <Wire.h>

// Variables definition
// HC05, bluetooth module
SoftwareSerial comBA(2,3);
String c = ""; //receive serial/bluetooth

// KY035, Hall sensor
#define ky035Signal 4
int ky035Read = 0;
int ky035Cont = 0;
double bikeDistance = 0.00;
double bikeSpeed = 0.00;
#define arduinoLedPin 13
int arduinoLedState = 0;

// MQ7, gas sensor
#define mq7AOUT A0
#define mq7DOUT 5
double mq7Value = 0.00;
double mq7Limit = 0.00;
double envGas = 0.00;
String mq7Range = "";

// LM35, temperature sensor
#define lm35Vs A1
double envTemperature = 0.00;
```

```
29 double lm35Voltage = 0.00;
30
31 // LDR, light sensor
32 #define ldrPin A2
33 double ldrValue = 0.00;
34 double envBrightness = 0.00;
35 String envBRange = "";
36
37 // Motor control
38 #define mcP0 6
39 #define mcP1 7
40 #define mcP2 8
41 #define mcP3 9
42 #define mcV0In A4
43 #define mcV0Out 10
44 double mcV0 = 0.00;
45 double bikeSpeedControl = 0.00;
46
47 // Battery lvl indicator
48 #define batPin A3
49 double batR1 = 10000.00;
50 double batR2 = 1000.00;
51 double batRawValue = 0.00;
52 double batVoltage = 0.00;
53 double batPercentage = 0.00;
54
55 // Function headings
56 void androidCom();
57 void hallSensor();
58 void coSensor();
59 void tempSensor();
60 void lightSensor();
61 void motorControl();
62 void batteryMon();
```

A.1.2 Função Setup

```
1 void setup(){
2     Serial.begin(9600);
3     comBA.begin(9600);
4
5     pinMode(ky035Signal,INPUT);
6     pinMode(arduinoLedPin,OUTPUT);
```

```
7
8       pinMode(mq7DOUT, INPUT);
9       pinMode(mq7AOUT, INPUT);
10
11      pinMode(lm35Vs, INPUT);
12
13      pinMode(ldrPin, INPUT);
14
15      pinMode(mcP0, OUTPUT);
16      pinMode(mcP1, OUTPUT);
17      pinMode(mcP2, OUTPUT);
18      pinMode(mcP3, OUTPUT);
19      pinMode(mcV0In, INPUT);
20      pinMode(mcV0Out, OUTPUT);
21
22      pinMode(batPin, INPUT);
23  }
```

A.1.3 Função Loop

```
1  void loop(){
2      androidCom();
3      hallSensor();
4      coSensor();
5      tempSensor();
6      lightSensor();
7      motorControl();
8      batteryMon();
9      delay(1000); //1s
10 }
```

A.1.4 Função Android Comunication

```
1  void androidCom() {
2      if(comBA.available()){  //speed + or
3          while (comBA.available()){
4              c += (char) comBA.read();
5              delay(10);
6          }
7          if(c.equals("on")){
8              bikeSpeedControl = 1;
9              ky035Cont = 0;
```

```
10          }
11          else if(c.equals("+1") && bikeSpeedControl>=1 &&
12          bikeSpeedControl<15) bikeSpeedControl++;
13          else if(c.equals(" 1") && bikeSpeedControl>1)
14          bikeSpeedControl  ;
15          else if(c.equals("off")){
16              bikeSpeedControl = 0;
17              ky035Cont = 0;
18          }
19          c = "";
20      }
21      String data = (String) bikeDistance + " " + (String) bikeSpeed +
22      " " + (String) mq7Range + " " + (String) envTemperature + " " +
23       (String) envBRange + " " + (String) bikeSpeedControl + " " +
24       (String) batPercentage;
25      comBA.println(data);
26  }
```

A.2 Códigos-fonte do Android

A.2.1 Parte do Manifest

```
1  android:allowBackup="true"
2  android:icon="@drawable/icon"
3  android:label="@string/app_name"
4  android:supportsRtl="true"
5  android:theme="@android:style/Theme.NoTitleBar">
6  <activity android:name=".SplashActivity">
7  <intent filter>
8  <action android:name="android.intent.action.MAIN" />
9
10 <category android:name="android.intent.category.LAUNCHER" />
11 </intent filter>
12 </activity>
13 <activity android:name=".MainActivity" />
14 <activity android:name=".HelpActivity" />
15 <activity android:name=".LogActivity" />
16 <activity
17 android:name=".DataActivity"
18 android:label="@string/title_activity_data" />
19
20 <meta data
21 android:name="com.google.android.geo.API_KEY"
```

```
22 android:value="@string/google_maps_key" />
23
24 <activity
25 android:name=".StartActivity"
26 android:label="@string/title_activity_start"></activity>
27
28 <meta data
29 android:name="com.google.android.gms.version"
30 android:value="@integer/google_play_services_version" />
```

A.2.2 Função Data Intent Change da activity Main

```
1 Intent i = new Intent(this,DataActivity.class);
2 startActivity(i);
```

A.2.3 Parte da função On Location Changed da activity Start

```
1 if (mk != null) mk.remove();
2 LatLng cyclist = new LatLng(location.getLatitude(),
3 location.getLongitude());
4 MarkerOptions mko = new MarkerOptions();
5 mko.position(cyclist);
6 mko.title("Here you are!");
7 mko.icon(BitmapDescriptorFactory.defaultMarker
8 (BitmapDescriptorFactory.HUE_YELLOW));
9 mk = mMap.addMarker(mko);
```

A.2.4 Parte da função Download da activity Log

```
1 ArrayList<String> out = Database.recoverDatabaseToString();
2
3 File dir = Environment.getExternalStoragePublicDirectory
4 (Environment.DIRECTORY_DOWNLOADS);
5 File f = new File(dir,"log.txt");
6 if (f.exists()) f.delete();
7 if (!f.exists()) f.createNewFile();
8
9 FileOutputStream stream = new FileOutputStream(f);
10 try {
11 for (int l=0;l<out.size();l++) {
12 stream.write(out.get(l).getBytes());
```

```
13 }
14 stream.close();
15 } catch (Exception e) {
16 e.printStackTrace();
17 }
```

A.2.5 Função Insert Database da classe Database

```
1     public static void insertDatabase(String day, String timeOn,
2     String timeOff, String distanceTotal, double distanceOff,
3     double speedOn, double speedOff, int week){
4     if (Double.isNaN(speedOff)) speedOff = 0f;
5     if (Double.isNaN(speedOn)) speedOn = 0f;
6     Log.d("qqInsert",day + "   " + timeOn + "   " + timeOff + "   "
7     + distanceTotal + "   " + distanceOff + "   " + speedOn + "   "
8     + speedOff + "   " + week);
9     sqDb.execSQL("INSERT INTO tab VALUES(" + day + "," + timeOn + ","
10    + timeOff + "," + String.format("%.2f",Double.valueOf
11    (distanceTotal)) + "," + String.format("%.2f",Double.valueOf
12    (distanceOff)) + "," + String.format("%.2f",Double.valueOf
13    (speedOn)) + "," + String.format("%.2f",Double.valueOf
14    (speedOff)) + "," + week + ");");
15    }
```

A.2.6 Parte da Splash Screen

```
1 super.onCreate(savedInstanceState);
2 setContentView(R.layout.activity_splash);
3
4 new Handler().postDelayed(new Runnable(){
5 @Override
6 public void run() {
7 /* Create an Intent that will start the Menu Activity. */
8 Intent mainIntent = new Intent(SplashActivity.this,MainActivity.class
9 SplashActivity.this.startActivity(mainIntent);
10 SplashActivity.this.finish();
11 }
12 }, SPLASH_DISPLAY_LENGTH);
```

A.2.7 Parte da activity Help

```
1 super.onCreate(savedInstanceState);
2 setContentView(R.layout.activity_help);
3
4 TextView helpTextView = (TextView) findViewById(R.id.helpTextView);
5
6 String helpString = "";
7
8 helpTextView.setText(helpString);
9 helpTextView.setTypeface(MainActivity.tf);
```