UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

ARTHUR PORTO BORGES

DESENVOLVIMENTO DE UM SISTEMA *HOLTER* DE ELETROCARDIOGRAMA

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2018

ARTHUR PORTO BORGES

DESENVOLVIMENTO DE UM SISTEMA *HOLTER* DE ELETROCARDIOGRAMA

Trabalho de Conclusão de Curso, apresentado à disciplina de Trabalho de Conclusão de Curso 2 – TCC2, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELN - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Eduardo Giometti Bertogna

CAMPO MOURÃO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

Desenvolvimento de Sistema *Holter* de Eletrocardiograma

por

Arthur Porto Borges

Trabalho de Conclusão de Curso apresentado no dia 19 de Novembro de 2018 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

Prof. Marcio Rodrigues da Cunha
(UTFPR)

_____

Prof. Roberto Ribeiro Neli
(UTFPR)

_____

Prof. Eduardo Giometti Bertogna
(UTFPR)
Orientador

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

AGRADECIMENTOS

Sou grato a minha mãe e meu pai, que não mediram esforços para me manter aqui durante os anos da graduação, e também aos meus irmãos Hermano e Lorenzo por tudo que representam em minha vida. Amo vocês.

Agradeço também o professor Eduardo Giometti Bertogna por toda a orientação disponibilizada para o desenvolvimento desse trabalho, por me fornecer os equipamentos necessários, me auxiliando sempre que me foi necessário. Agradeço também todos os professores que estiveram presentes durante minha graduação, pois também são responsáveis pelo conhecimento adquirido até aqui e também todos os funcionários da instituição.

Sou grato aos meus colegas de sala pela ajuda e troca de informações que me ofereceram quando necessário em sala de aula e também aos amigos que fiz durante o período em que estive fora de casa, por todo auxílio em momentos felizes e difíceis que passei fora.

RESUMO

O presente trabalho tem por objetivo apresentar um sistema *holter* de eletrocardiograma (ECG), construído a partir de dispositivos de baixo custo, a partir de propostas já existentes no mercado como base para elaboração do mesmo. Este tipo de sistema é utilizado para coletar longos períodos da atividade elétrica do coração, a fim de buscar momentos que não puderam ser analisados em um exame comum, que leva cerca de alguns poucos minutos. Foram elencados dispositivos que poderiam ser utilizados para a aplicação, bem como as plataformas de desenvolvimento para o trabalho. O sinal utilizado para a aplicação, foi retirado do MIT- BIH utilizando um microcontrolador e um conversor digital-analógico (DAC), como simulador. O software foi desenvolvido por meio da IDE Arduino, por afinidade. Os resultados são armazenados em arquivos, dos quais serão lidos com *Virtual Instrucment* (VI) desenvolvido para a aplicação, baseado em LabView. O dispositivo é alimentado através de baterias de lítio de 3.7V, associadas em série, por se tratar de um dispositivo portátil. Os sinais coletados pelo dispositivo *Holter*, representaram com fidelidade o sinal gerado pelo simulador desenvolvido para a aplicação, representado no VI.

Palavras Chave: Holter de ECG, Eletrocardiografia, Microcontroladores.

ABSTRACT

The objective of the present study is to present a holter electrocardiogram (ECG) system, built from low cost devices, based on proposals already on the market as a basis for the elaboration of the same. This type of system is used to collect long periods of the electrical activity of the heart in order to look for moments that could not be analyzed in a common examination, which takes about a few minutes. Devices that could be used for the application as well as development platforms for the job were listed. The signal used for the application was removed from the MIT-BIH using a microcontroller and a digital-to-analog converter (DAC) as a simulator. The software was developed through the IDE Arduino, by affinity. The results are stored in files, which will be read with Virtual Instruction (VI) developed for the application, based on LabView. The device is powered by 3.7V lithium batteries, associated in series, as it is a portable device. The signals collected by the Holter device, represented with fidelity the signal generated by the simulator developed for the application, represented in the VI.

Key words: ECG Holter, Eletrocardiography, Microcontrollers.

LISTA DE FIGURAS

Figura 1: Direção do fluxo sanguíneo. 11
Figura 2: Funcionamento da Bomba de Sódio e Potássio. 12
Figura 3: Onda PQRST resultante. 14
Figura 4: Derivações Bipolares. 15
Figura 5: Derivações aumentadas. 16
Figura 6: Derivações precordiais. 16
Figura 7: Diagrama de blocos da obtenção dos biopotenciais. 19
Figura 8: Diferentes faixas de frequência do exame ECG. 20
Figura 9: Espectro do sinal de ECG. 21
Figura 10: Filtro Passa-faixa, sendo um FPB o primeiro estágio. 21
Figura 11: Visão superior do kit de desenvolvimento escolhido. 26
Figura 12: Esquemático AD8232. 27
Figura 13: Circuito da configuração de Monitor Cardíaco. 28
Figura 14: Placa de aquisição de sinais. 28
Figura 15: Resposta em frequência do monitor cardíaco. 29
Figura 16: Esquemático da simples fonte de tensão. 30
Figura 17: Esquemático do sistema. 30
Figura 18: Módulo cartão de memória. 31
Figura 19: Fluxograma base para as atividades. 34
Figura 20: Fluxograma da rotina de configuração de Hora/Data. 36
Figura 21: Fluxograma de atividade. 37
Figura 22: Fluxograma para início das gravações. 38
Figura 23: Fluxograma da rotina de gravação. 39
Figura 24: Estado de inicialização. 40
Figura 25: Estado de parada. 41
Figura 26: Evento para finalização da aplicação. 42
Figura 27: Evento para abrir arquivo. 43
Figura 28: Evento de seleção da taxa de amostragem. 43
Figura 29: Protótipo de aquisição de sinais. 45
Figura 30: Simulador de ECG. 46
Figura 31: Sinal de saída do simulador. 47

Figura 32: Arquivos gerados no cartão de memória.....48
Figura 33: Consumo de corrente do simulador. 49
Figura 34: Consumo de corrente do holter.....50
Figura 35: Layout do software desenvolvido para leitura das amostras.....51
Figura 36: Sinal com amostras coletadas pelo protótipo.....52
Figura 37: Mensagem por falta de cartão SD.....53
Figura 38: Tela de Ajuda. 53
Figura 39: Tela do menu MOSTRA. 54
Figura 40: Tela para início das gravações 54
Figura 41:Tela durante gravação 55

LISTA DE TABELAS

Tabela 1: Derivações mais utilizadas na prática pelo holter.....17
Tabela 2: Especificações de tecnologias existentes no mercado mundial. 18
Tabela 3: Valor dos dispositivos.....56

ABREVIATURAS, SIGLAS E ACRÔNIMOS

ADC	*Analog to Digital Converter* (Conversor Analógico Digital)
Ah	Ampér hora
CI	Circuito Integrado
DAC	*Digital to Analog Converter* (Conversor Digital Analógico)
DCI	Doença Cardíaca Isquêmica
ECG	Eletrocardiógrafo
FPA	Filtro Passa Baixas
FPB	Filtro Passa Altas
IAM	Infarto Agudo do Miocárdio
IDE	*Integrated Development Enviroment* (Ambiente de Desenvolvimento Integrado)
IHM	Interface Homem Máquina
I2C	*Inter-Integrated Circuit* (Circuito Inter-integrado)
Hz	Hertz
OLED	*Organic Light Emissor Diode* (Diodo Orgânico Emissor de Luz)
RTC	*Real Time Clock* (Relógio de Tempo Real)
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory* (Memória Estática de Acesso Aleatório)
USB	*Universal Serial Bus* (Barramento Serial Universal)
V	Volts

SUMÁRIO

1 - INTRODUÇÃO 7
1.1 - HISTÓRICO 7
1.2 - JUSTIFICATIVA 8
1.3 - OBJETIVO 9
2 - FUNDAMENTAÇÃO TEÓRICA 10
2.1 - O CORAÇÃO 10
2.2 - ELETROFISIOLOGIA CELULAR 11
2.3 - ELETROCARDIOGRAMA 13
2.4 – DERIVAÇÕES PADRÃO 14
3 – PROPOSTA DE SISTEMA 17
3.1 - ANÁLISE DE SISTEMAS EXISTENTES 17
3.2 – AQUISIÇÃO DO SINAL DE ECG 18
3.3 – ELETRODOS 19
3.4 – AMPLIFICADOR DE INSTRUMENTAÇÃO 19
3.5 – FILTRO PASSA-FAIXA 20
3.6 – PROCESSAMENTO DO SINAL E ARMAZEMENTO DOS DADOS 21
4 – PLATAFORMAS DE DESENVOLVIMENTO 24
4.1 – ALTERNATIVAS PARA AS PLATAFORMAS DE *HARDWARE* 24
4.2 – ALTERNATIVAS PARA AS PLATAFORMAS DE *SOFTWARE* 24
5 - MATERIAIS E MÉTODOS 26
5.1 – PLACA DE DESENVOLVIMENTO 26
5.2 – PLACA DE AQUISIÇÃO DE SINAIS DE ECG 27
5.3 – FONTE DE ALIMENTAÇÃO 29
5.4 – PLACA DO CARTÃO DE MEMÓRIA 31
5.5 – INTERFACE HOMEM MÁQUINA 31
5.6 - TIMER E CONVERSOR ANALÓGICO DIGITAL 32
5.7 – FLUXOGRAMA DA ROTINA PRINCIPAL 33
5.8 – FLUXOGRAMA DA CONFIGURAÇÃO DE HORA/DATA 35
5.9 – FLUXOGRAMA DA ROTINA DE GRAVAÇÃO 37
5.10 – SOFTWARE PARA LEITURA DAS AMOSTRAS 39
5.11 - O PROTÓTIPO 44
5.12 - O SIMULADOR 45

6 – RESULTADOS.....47
6.3 - DO ARQUIVO DE MOMENTOS COLETADOS 48
6.4 - DA FONTE DE ALIMENTAÇÃO DOS MICROCONTROLADORES 49
6.5 - DO *VIRTUAL INSTRUMENT*.....50
6.6 – DA APLICAÇÃO 52
7 – CONCLUSÕES.....55
7.1 – TRABALHOS FUTUROS.....56
REFERÊNCIAS.....57
ANEXO A – ALGORITMO DESENVOLVIDO PARA A APLICAÇÃO 60

1 - INTRODUÇÃO

Para melhor entendimento em torno do tema abordado, este capítulo traz informações acerca da contextualização histórica, as justificativas sobre a escolha do tema, os objetivos a serem alcançados e cronograma das atividades do projeto.

1.1 - HISTÓRICO

Após o término da Segunda Guerra Mundial, e com o início da Guerra Fria que separou o mundo em dois lados, um socialista e outro capitalista, deflagrou-se um grande avanço tecnológico em todas as áreas do conhecimento humano, como a Engenharia Biomédica que usa as ciências exatas e conhecimento de engenharia na área da saúde, iniciando seus estudos nos sistemas biológicos complexos (Bioengenharia) e Engenharia de Reabilitação. Outras duas áreas da Engenharia Biomédica também entraram em ação com o passar dos anos, sendo a Engenharia médica responsável pelo desenvolvimento de dispositivos para uso médico, e a engenharia clínica, que trata da gestão hospitalar (Engenharia Biomédica, 2017). Tais avanços proporcionaram um aumento na expectativa de vida mundial auxiliando na prevenção, diagnóstico e tratamento de doenças.

Assim, para obter o diagnóstico de possíveis doenças cardíacas e cardiovasculares, que assola grande parte da população mundial, algumas tecnologias começaram a ser elaboradas e desenvolvidas a fim de melhorar a vida do ser humano. Como relata Fye (1994), citado por Giffoni e Torres (2010) que após a segunda metade do século XX, a eletrofisiologia cardiovascular teve bastante desenvolvimento, evidenciando monitoramento por *holter,* estudos eletrofisiológicos invasivos, eletrocardiograma de alta resolução e mapeamento intracardíaco.

O monitoramento eletrocardiográfico ambulatorial (*holter*), recebeu o nome de seu inventor, o biofísico Norman Jefferis Holter, que se tornou cada vez mais avançado desde sua invenção, no início da década de 1960.

Os dispositivos se aperfeiçoaram muito desde então, tornando-se cada vez mais leves, compactos, e mais confiáveis devido o surgimento da tecnologia digital, que possibilitou eliminar distorções antes presentes. O registro de *holter* típico apresenta duração de 24 a 48 horas, possibilitando a detecção de anormalidades

eletrocardiográficas transitórias no contexto de vida normal do paciente (GIFFONI e TORRES, 2010).

Com este dispositivo o sinal de eletrocardiograma (ECG) pode ser registrado, analisado por profissionais devidamente qualificados para verificar possíveis variações, diagnosticar algumas cardiopatias, e então tratá-las devidamente.

1.2 - JUSTIFICATIVA

Segundo a Organização Mundial da Saúde (OMS), a doença cardiovascular é a maior causadora de mortes no mundo. Em 2015, cerca de 17,7 milhões de pessoas morreram de doença cardiovascular, representando 31% de todas as mortes do mundo, ocupando o topo da lista de cardiopatia, a doença cardíaca isquêmica (DCI) (WHO, 2017).

As arritmias cardíacas têm lugar de destaque entre os distúrbios funcionais cardiovasculares, e são definidas como alterações de formação e condução do estímulo elétrico, podendo ser benigna, que se manifesta em indivíduos sem doença estrutural do coração, ou maligna, que pode levar a morte súbita. No infarto agudo do miocárdio (IAM), manifestação clínica mais comum da DCI, a morte acontece em 50% dos casos nas primeiras duas horas, sendo causada por arritmias cardíacas de origem ventricular, em especial extra-sístoles ventriculares (ESV) e taquicardia ventricular (TV), que degeneram em fibrilação ventricular (FV) levando à parada cardiorrespiratória (PCR) (BERTONHA, 1993).

O diagnóstico das arritmias cardíacas pode ser obtido através do ECG, porém o exame de repouso dura poucos minutos, e a maioria das arritmias têm caráter transitório, objetivando o monitoramento por longos períodos, realizado pelo dispositivo *holter.* O uso desta tecnologia propiciou a identificação do mecanismo das arritmias no pós-IAM e a importância do diagnóstico da instabilidade elétrica do coração e de suas possíveis consequências para o indivíduo. Outras aplicações para o método seriam na investigação de palpitações, síncope e pré-síncope, taquicardias, dor torácica, controle terapêutico com drogas antiarrítmicas e controle de marca-passo cardíaco artificial (BERTONHA, 1993).

O *holter* utilizava fita magnética como meio de armazenamento de dados, porém incertezas eram adicionadas ao sinal obtido devido aos problemas associados

ao uso de partes móveis sujeitas a desgaste e rotação irregular da fita. Assim, este método se tornou obsoleto com a evolução da tecnologia digital em meados dos anos 90, que proporcionou um armazenamento em memórias não-voláteis, e o uso de microcontroladores tornando a aquisição de dados melhor e mais confiável.

Diante da importância de se utilizar o *holter* para obter o diagnóstico de arritmias cardíacas, e dos elevados custos destes dispositivos, em sua maioria importados, evidencia a necessidade de se desenvolver um equipamento que seja mais acessível.

1.3 - OBJETIVO

O objetivo deste trabalho é o desenvolvimento do projeto de um *holter* de ECG com capacidade de monitoramento por 24 horas, oferecendo uma aquisição e processamento de sinais com confiabilidade através de tecnologias disponíveis no mercado afim de se tornar um dispositivo com otimização do dimensionamento e custo reduzido.

As etapas a serem alcançadas são:

- Analisar o funcionamento, a atividade elétrica do coração, que é verificado através do eletrocardiograma.

- Analisar os dispositivos existentes no mercado nacional e internacional, a fim de elaborar um projeto seguindo especificações similares.

- Desenvolver um protótipo de tamanho reduzido e com baixo consumo de energia.

- Analisar tecnologias disponíveis para realizar o processamento e armazenamento dos sinais.

- Desenvolver um software capaz de ter uma boa interação em homem e a máquina, em si tratando de um protótipo base para um produto final.

- Obter resultados satisfatórios em relação a gravação das amostras, para que o sinal seja representado com fidelidade.

2 - FUNDAMENTAÇÃO TEÓRICA

Serão abordados neste capítulo conceitos sobre o coração, seu funcionamento, o registro da atividade elétrica e como capta-la.

2.1 - O CORAÇÃO

O coração é o principal órgão do sistema circulatório, localizado na caixa torácica com tamanho aproximado de um punho fechado, com 2/3 da sua massa a esquerda da linha média do corpo e é capaz de bombear sangue a todas as extremidades do corpo, realizando o transporte de oxigênio e nutrientes para todas as células que sustentam as atividades orgânicas (AULA DE ANATOMIA, 2017).

É um músculo oco, composto em sua maior parte pelo miocárdio, envolvido externamente pelo pericárdio, que mantem o coração na posição e fornece proteção, e o endocárdio que é uma membrana que reveste as quatro cavidades internas do coração, sendo duas superiores chamadas de átrios e as inferiores, ventrículos.

Os átrios e ventrículos são divididos verticalmente pelo septo interatrial e interventricular e horizontalmente pelas válvulas atrioventriculares, sendo a mitral ou bicúspide localizada ao lado esquerdo arterial e a válvula tricúspide se encontra do lado direito. Também existem as válvulas aórtica e pulmonar, sendo as quatro responsáveis por manter o sangue nas câmaras por determinado tempo, para conferir pressão e evitar o refluxo sanguíneo (GUYTON e HALL, 2006).

O sangue proveniente dos tecidos humanos, rico em gás carbônico (CO_2) entra no átrio direito, após uma estimulação, seguindo para o ventrículo direito que o direcionará para o pulmão, para realizar a troca gasosa, retornando com sangue arterial ou oxigenado até o átrio esquerdo, que segue ao ventrículo direito que enviará todo este sangue ao resto do corpo novamente. A Figura 1 apresenta a direção em que o sangue percorre o coração.

Figura 1: Direção do fluxo sanguíneo.

Fonte: (GUYTON e HALL, 2006).

O ciclo cardíaco é composto de uma contração ventricular conhecida por sístole e um relaxamento ventricular, em que o coração se enche de sangue denominado diástole. O ciclo tem seu início a partir de um estímulo no nodo sinusal, este é espalhado pelas fibras musculares especializadas excitatórias e condutoras.

2.2 - ELETROFISIOLOGIA CELULAR

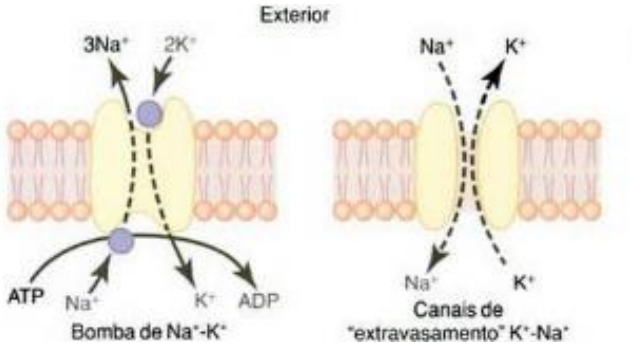
Todos os potenciais bioelétricos no corpo humano são gerados por células nervosas ou musculares, e têm origem iônica. Estas células possuem uma membrana plasmática que é semipermeável e funciona como um filtro, que controla principalmente a concentração dos íons potássio (K^+), sódio (Na^+) e cloreto (Cl^-), deixando o meio extracelular mais rico em sódio e cloreto, ao passo que o meio intracelular se torna mais rico em potássio e algumas proteínas (TORTORA e DERRICKSON, 2017).

A membrana celular tem um papel muito importante para a vida humana, sendo capaz de realizar o transporte de substâncias por meio de difusão, ou ativamente. Caso somente o primeiro acontecesse, as diferentes concentrações iônicas igualar-se-iam, acarretando em um inchamento de todas as células humanas,

mas algumas proteínas presentes na membrana oferecem vias alternativas para substâncias serem transportadas de forma ativa. A bomba de sódio e potássio é um exemplo de transporte ativo, em que ocorre o transporte de substâncias contra o gradiente de concentração, mantendo a diferença de concentração entre os meios e estabelecer uma eletronegatividade em seu interior (GUYTON e HALL, 2006).

Através da proteína transportadora ATP (adenosina trifosfato) dois íons de potássio se ligam a parte externa da célula, enquanto três íons de sódio se ligam a parte interna. Assim a quebra da molécula em ADP (adenosina difosfato) ocorre, liberando energia necessária para jogar os íons de sódio e potássio contra o gradiente de concentração, como representa a **Erro! Fonte de referência não encontrada.**.

Figura 2: Funcionamento da Bomba de Sódio e Potássio.



Fonte: (GUYTON e HALL, 2006).

A bomba de sódio e potássio está diretamente ligada aos processos de contração muscular e condução dos impulsos nervosos, além de facilitar a penetração de aminoácidos e açúcares. A manutenção da concentração de potássio no meio intracelular é importante para a síntese de proteína e respiração, e o bombeamento de sódio para fora da célula permite a manutenção do equilíbrio osmótico. Além disso, através deste transporte, ocorre a estabilidade do volume celular e a concentração de água no meio intracelular (SILVA, D., 2017).

Com o equilíbrio do gradiente de concentração e a força elétrica entre os meios intra e extracelulares uma diferença de potencial de aproximadamente -90 mV conhecida como potencial de repouso é estabelecida, e quando um estímulo é fornecido a célula, a condutância da membrana celular varia e causa a mudança no fluxo de corrente de carga positiva, iniciando o potencial de ação da célula, que se

inicia com o aumento elevado da condutância de sódio no primeiro momento afim de aumentar o potencial interno celular e chegar ao equilíbrio de corrente iônica (GUYTON e HALL, 2006).

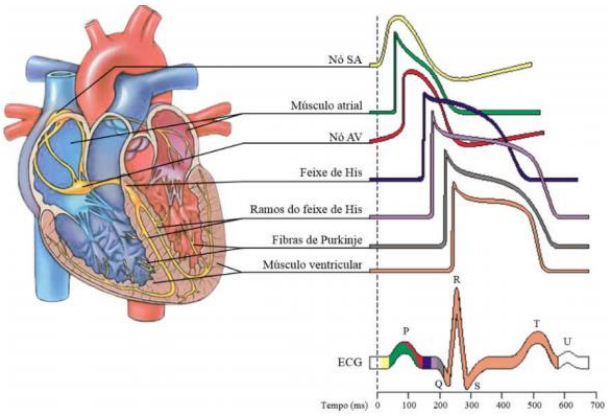
Quando este equilíbrio é alcançado o potencial interno da célula é de aproximadamente 20 mV, assim o potencial de ação tem sua despolarização, e a condutância de sódio volta ao normal enquanto a condutância de potássio aumenta, fazendo com que tenha um rápido movimento de potássio para fora da célula e retornando ao potencial de repouso, onde as concentrações iônicas são reestabelecidas através da bomba de sódio e potássio.

Uma ritmicidade dos potenciais de ação é apresentada em alguns tecidos, como músculos lisos, intestino, neurônios e músculo cardíaco. No caso dos potencias de ação gerados nas fibras musculares do coração, tem-se o platô. Este é caracterizado pelo atraso na despolarização da membrana, que leva de 0,2 a 0,3 s e está diretamente ligado ao fato de dois tipos de canais participarem da despolarização, os rápidos de sódio e os canais lentos de sódio e cálcio. A abertura dos canais de potássio, são mais lentas que o usual, que pode contribuir também para o platô verificado.

2.3 - ELETROCARDIOGRAMA

O registro verificado no sinal ECG, é resultado da somatória dos estímulos elétricos que passam pelas células miocárdicas, causando a despolarização destas, formando o traçado característico conhecido como PQRST, conforme a Figura 3.

Figura 3: Onda PQRST resultante.



Fonte: (MALMIVUO & PLONSEY, 1995) apud (TURCHIELLO, 2014).

A onda se inicia quando o marcapasso natural, também conhecido como nodo sinoatrial recebe um potencial de ação realizando a despolarização atrial, sendo a contração atrial caracterizada pela onda P (DEPARTAMENTO DE FISIOLOGIA E FARMACOLOGIA, 2006).

A seguir, a onda de despolarização continua seu caminho em direção ao nodo atrioventricular, o que leva cerca de 0,12 a 0,2 segundos, tempo necessário para o sangue passar pelas válvulas atrioventriculares e é representado no gráfico pelo intervalo PR. Durante o intervalo PR, a onda segue sua propagação através do feixe de His, que se ramifica pelos ventrículos esquerdo e direito e suas ramificações das fibras de Purkinge, causando a contração ventricular. A depolarização ventricular forma várias ondas, denominada pelo complexo QRS (RESENDE, COLANTONI, *et al.*, 2008) (ECGPEDIA, 2014).

O traçado do intervalo ST traz importantes informações acerca de possíveis doenças, e a onda T representa a repolarização ventricular ou o relaxamento dos ventrículos. A onda de repolarização atrial fica suprimida pela despolarização ventricular que tem potencial elétrico superior.

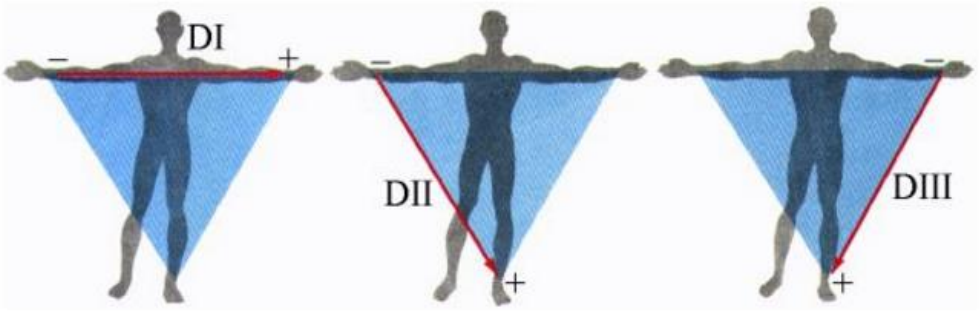
2.4 – DERIVAÇÕES PADRÃO

As ondas bioelétricas geradas pela ativação de diferentes regiões miocárdicas podem ser monitoradas na superfície da pele, captadas através de eletrodos posicionados de forma que se crie um vetor cardíaco através de um dipolo elétrico

para registrar a atividade do coração em direção e magnitude nos três planos de referência ortogonais.

O triângulo de Einthoven descreve as três derivações mais conhecidas através de um triângulo equilátero, usadas desde a obtenção do primeiro sinal de ECG. Este triângulo descreve as derivações bipolares, e tem eletrodos posicionados entre os braços direito e esquerdo formando a derivação DI com ângulo de orientação de 0º, entre braço direito e perna esquerda formando a derivação DII orientada a 60º e entre o braço e perna esquerda formando a derivação DIII orientada a 120º, conforme mostra a **Erro! Fonte de referência não encontrada.**.

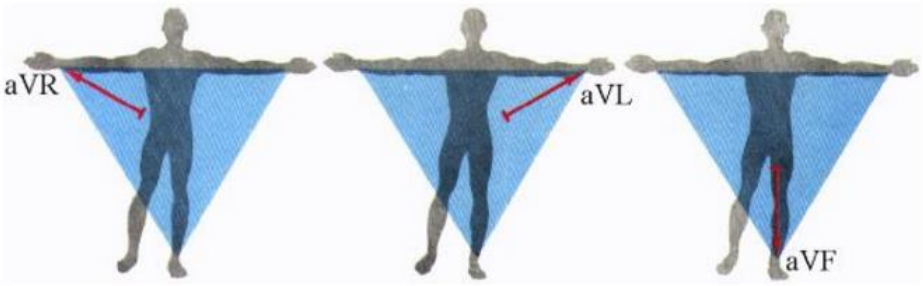
Figura 4: Derivações Bipolares.



Fonte: (DUARTE, 2017).

Outras três derivações padrão, são derivações unipolares, conhecidas como aumentadas, que usa um ponto de referência entre os vértices do triangulo de Einthoven e intercalados a uma resistência de 5000 Ohms que funciona como um eletrodo indiferente, visto que a soma vetorial das derivações dispostas em 60º no plano frontal é igual a zero. Desta forma é medido o potencial absoluto da região conectada como mostra a **Erro! Fonte de referência não encontrada.**.

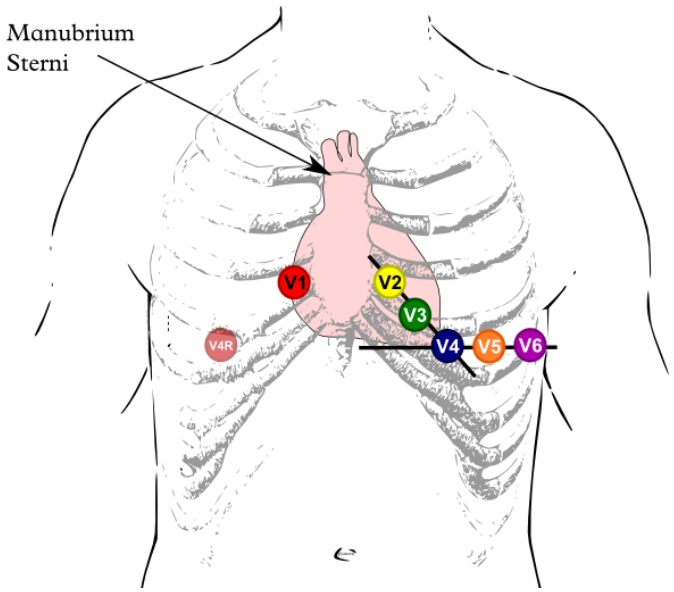
Figura 5: Derivações aumentadas.



Fonte: (DUARTE, 2017).

As seis derivações já vistas descrevem o vetor cardíaco no plano frontal, já as outras seis descrevem o vetor cardíaco no plano horizontal, são unipolares e conhecidas como precordiais, com eletrodos posicionados cuidadosamente na região do tórax conforme **Erro! Fonte de referência não encontrada.**.

Figura 6: Derivações precordiais.



Fonte: (ECGPEDIA, 2014).

A eletrocardiografia realizada em hospitais e clínicas realiza o monitoramento da atividade cardíaca através das 12 derivações padrão, enquanto o dispositivo *holter* realiza suas gravações utilizando-se duas ou três derivações bipolares precordiais, sendo a derivação CM5 a que possui maior sensibilidade para o diagnóstico de alterações do ritmo e a detecção de isquemia miocárdica. As derivações mais

utilizadas para a monitorização ambulatorial e o posicionamento dos eletrodos no tórax encontram-se detalhadas na Tabela 1 (GRUPI, BRITO e UCHIDA, 1999).

Tabela 1: Derivações mais utilizadas na prática pelo holter.

Derivação	Canal	Local
CM5	CH 1(-) CH 1(+)	clavícula direita, justa lateral ao esterno sobre a 5ª Costela, na linha axilar anterior esq.
CM1	CH 2(-) CH 2(+)	clavícula esquerda, justa lateral ao esterno sobre a 4ª articulação esterno-costal direita
Terra		últimos arcos costais à direita
CC5	CH 3(-) CH 3(+)	6ª costela, na linha axilar média direita 6ª costela, na linha axilar média esquerda
Para marca-passos	CH 2 (-) CH 2(+)	sobre o manúbrio do esterno sobre a 4ª costela, na linha médio clavicular.

Fonte: (GRUPI, BRITO e UCHIDA, 1999).

3 – PROPOSTA DE SISTEMA

Este capítulo cita alguns sistemas existente e a proposta do sistema a ser desenvolvido.

3.1 - ANÁLISE DE SISTEMAS EXISTENTES

Foi realizado uma pesquisa sobre algumas tecnologias existentes no mercado afim de coletar especificações de projeto para que o protótipo a ser desenvolvido com similaridade, com a finalidade de obter um dispositivo de baixo custo, quando comparado com sistemas existentes, que tem valor médio de mercado de R$800,00. Algumas especificações das tecnologias analisadas, são apresentadas na Tabela 2.

Tabela 2: Especificações de tecnologias existentes no mercado mundial.

Especificaçoes	Dispositivos			
	Heal Force Prince-180B	Mortara Instrumet H3+	Biomedical Systems BMS300	Nasiff H1
Número de canais	1	2 ou 3	3	3
Faixa de Frequência (Hz)	0,5 a 40	0,05 a 60	0,5 a 40	0,05 a 40
Taxa de amostragem (amostras/s)	(-)	180	128, 256, 512 e 1024	(-)
Tempo de registro (h)	>24	>48	96, 72, 48 e 24	24
CMRR (dB)	60	(-)	60	(-)
Resolução (bits)	(-)	12	10	(-)
Display	LCD	LCD	OLED	LCD
Armazenameto de dados	SD card	Interna	SD card	Interna
Massa (g)	106	28	114	283,5

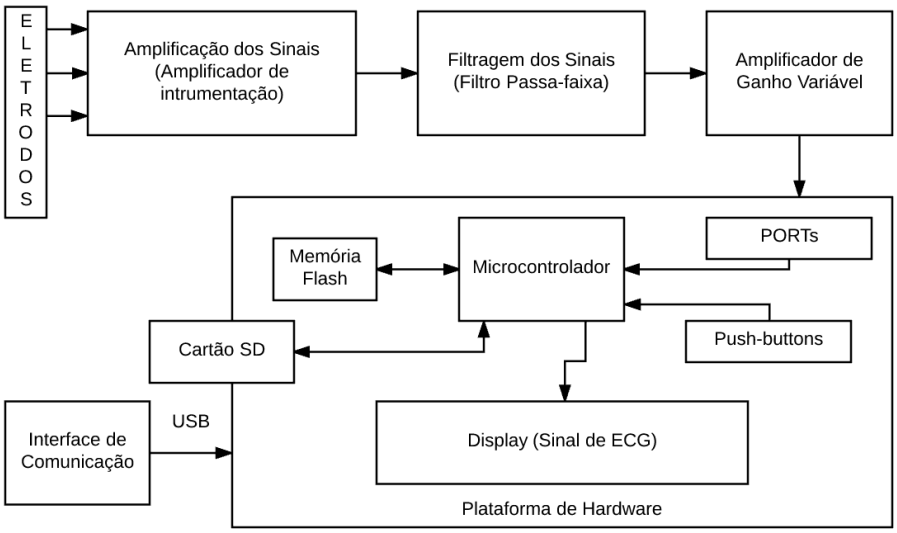
Fonte: Autoria Própria.

3.2 – AQUISIÇÃO DO SINAL DE ECG

Até o momento, é inquestionável a importância de entender os sinais bioelétricos para estudo dos sistemas biológicos e promover melhoras à vida humana. Para realizar a aquisição destes biopotenciais, faz-se necessário a criação de um circuito eletrônico capaz de captar o sinal elétrico na superfície da pele, que tem baixa amplitude (da ordem de alguns mV) e com alta qualidade.

Sendo assim, tem-se a necessidade de reduzir a interferência entre pele e eletrodo, amplificar o sinal para melhor compreensão, filtrar sinais indesejados, como artefatos de respiração e ruído muscular, e então processá-lo. Para isso, a Figura 7 representa em forma de diagrama de blocos as etapas para condicionamento destes sinais.

Figura 7: Diagrama de blocos da obtenção dos biopotenciais.



Fonte: Autoria própria.

A interface de comunicação é um meio de se obter os resultados dos exames realizados, e os botões push-buttons são para registrar desconfortos do paciente durante o exame para posterior análise. O microcontrolador é o responsável por processar o sinal obtido.

3.3 – ELETRODOS

Uma das dificuldades na aquisição do sinal é o contato entre a pele e o eletrodo, usualmente não-invasivos para a presente aplicação. Estes eletrodos são compostos de prata/cloreto de prata (Ag/AgCl), que ameniza os níveis de ruído pelo fato de ser um eletrodo não polarizável, facilitando o fluxo de corrente pela interface. Para melhor captação do sinal bioelétrico, recomenda-se reduzir a impedância da pele no contato do eletrodo, realizando a tricotomia da região e limpeza com álcool para retirar células mortas.

3.4 – AMPLIFICADOR DE INSTRUMENTAÇÃO

Buscando obter um sinal de ECG de qualidade, o uso de um amplificador operacional de alta performance na saída do bloco dos eletrodos se faz necessário devido os seus importantes atributos para a aplicação. O circuito integrado proposto foi o INA128 da Texas Instruments®, e é totalmente indicado em aplicações de instrumentação médica, aquisição de sinais diversos e tem as seguintes caraterísticas importantes para o sistema, descritas a seguir:

-Alta relação de rejeição de modo comum (120 dB min).

-Alta impedância de entrada.

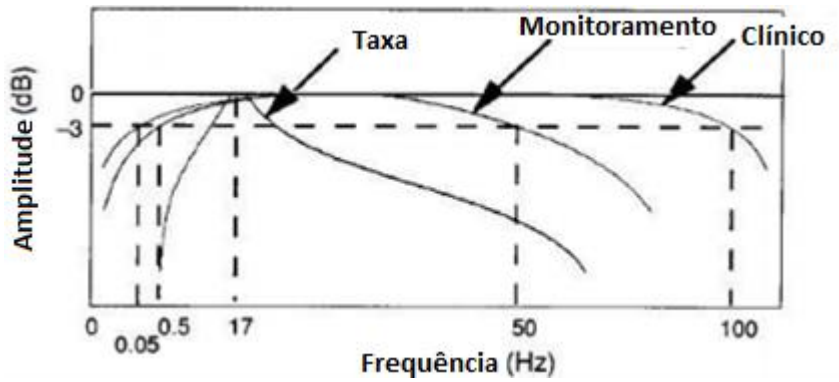
-Baixo consumo (700μA).

-Ganho variável de 1, 10, 100 e 1000.

3.5 – FILTRO PASSA-FAIXA

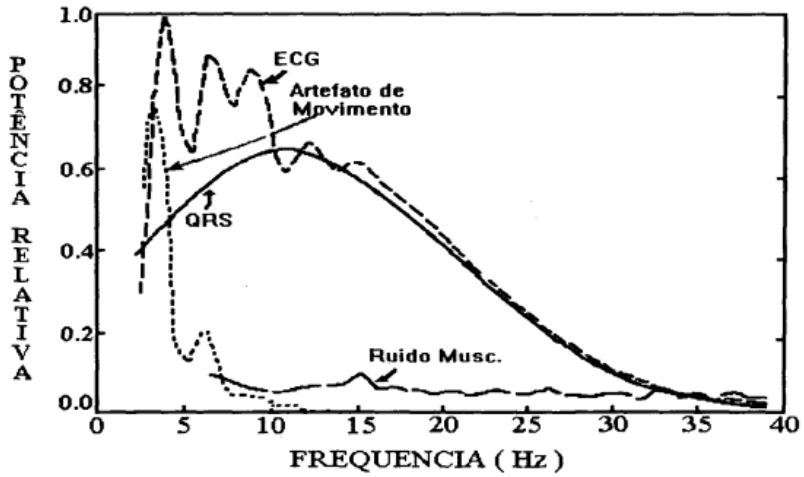
Em meio as diferentes tecnologias de análise do ECG, a banda de transição entre esses aparelhos se difere, sendo que em aplicações clínicas, onde se realiza o ECG de repouso a faixa está compreendida entre 0,05 a 100 Hertz (Hz), já nos dispositivos *holter* é de 0,5 a 50 Hz, visto que a principal componente da frequência cardíaca está em 17 Hz (TOMPKINS, 1993). O filtro implementado para esta aplicação, segue o padrão de modelos de eletrocardiógrafos ambulatoriais comerciais, os quais apresentam o sinal com banda de transição compreendidas entre 0,5 e 40 Hz. A Figura 8 e 9, apresentam características do sinal de ECG em frequência.

Figura 8: Diferentes faixas de frequência do exame ECG.



Fonte: (TOMPKINS, 1993).

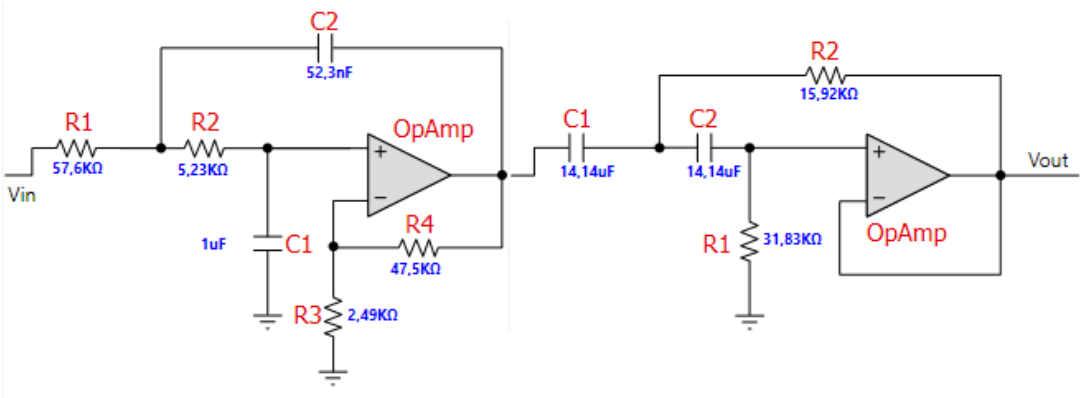
Figura 9: Espectro do sinal de ECG.



Fonte: (BERTONHA, 1993).

A filtragem será realizada utilizando dois filtros em cascata, sendo um filtro passabaixas (FPB) e um passa-altas (FPA), ambos do tipo *Butterworth* de ordem 2, topologia *Saleen-key*, e ganho de 20 V/V no FPB. O projeto dos filtros foi executado a partir do software gratuito da *Texas Instruments*®FilterPro, com valores de entrada de 0,5 Hz para FPA e de 40 Hz para FPB, e tem esquemático representado na Figura 10.

Figura 10: Filtro Passa-faixa, sendo um FPB o primeiro estágio.



Fonte: Autoria própria.

3.6 – PROCESSAMENTO DO SINAL E ARMAZEMENTO DOS DADOS

Após as etapas de condicionamento do sinal, no qual sinais indesejados são atenuados e as informações de interesse são melhoradas, faz-se o processamento

digital do sinal, afim de manipular ou transformar as informações presentes no sinal. Esta técnica consiste em representar um sinal de natureza contínua pelos seus valores de amplitude instantâneos tomados em pontos periódicos no tempo, para que o mesmo possa ser processado numericamente utilizando um hardware digital (TURCHIELLO, 2014).

Para isso, o sinal contínuo passa por um conversor analógico/digital (ADC), e resulta em uma sequência de amostras que podem ser processadas em tempo real ou armazenadas. Este processo é denominado digitalização e é dividido em duas etapas: a amostragem, onde o sinal é representado por uma série de amostras e a quantização, que atribui o valor discretizado da amplitude de cada amostra.

O processo da amostragem, consiste em transformar um sinal contínuo em um sinal discreto utilizando um número *n,* inteiro que representa o número da amostra do sinal e se relaciona no tempo a partir da equação 1:

$$x_a(t) \rightarrow x(nTs) \qquad \text{Equação 1.}$$

Onde Ts é um intervalo de tempo conhecido e constante chamado período de amostragem. O inverso do período de amostragem é a taxa de amostragem ou frequência de amostragem (Fs), dada em Hertz (TURCHIELLO, 2014).

Neste processo, deve ser respeitada a taxa de Nyquist, onde a frequência de amostragem deve ser superior ao dobro da frequência de corte do sinal, para que o mesmo não sofra *aliasing.*

Já a quantização consiste em representar os valores de amplitude do sinal contínuo, no tempo discreto a partir de um conjunto de bits. O número de bits de um conversor está relacionado com sua resolução. Para um conversor de 8bits e tensão de referência de 5 V a resolução é a mínima variação de tensão detectada pelo conversor, produzindo variação no código digital, e será: $5/2^8 \approx 0{,}01953$ V. Para um conversor de 12 bits, nas mesmas condições, a resolução passa a ser ≈ 0,001221 V. Assim, quanto maior o número de bits do ADC mais fiel será a digitalização e menor será o erro de quantização (TURCHIELLO, 2014).

As especificações escolhidas para o projeto, em relação ao processamento do sinal de ECG, com base na análise de sistemas existentes são de: resolução de 12 bits e taxa de amostragem de 240 Hz. Assim, estima-se que a quantidade de dados

gerados durante a aquisição feita através de 2 canais de ECG durante 24 horas é de: 24 horas x 3600 segundos x 240 Hz x 2 Bytes 2 canais = 82,94 MB.

Outra etapa se faz necessária caso a escolha da tecnologia abordada não tenha capacidade de armazenar a quantidade de dados adquiridos, denominada compressão, que tem propósito fundamental de reduzir o número de bits por amostra, representadas com fidelidade. Existem algumas técnicas de compressão do sinal digital e alguns aspectos devem ser levados em conta como o desempenho da compressão, a qualidade dos sinais reconstruídos, a complexidade em relação a esforço computacional e retardo na comunicação referente a complexidade do método.

4 – PLATAFORMAS DE DESENVOLVIMENTO

As plataformas pesquisadas para realizar o desenvolvimento do projeto são apresentadas neste capítulo, sendo listados alguns *hardwares* e *softwares.*

4.1 – ALTERNATIVAS PARA AS PLATAFORMAS DE *HARDWARE*

Seguindo especificações do sistema proposto, alguns requisitos de *hardware* são requeridos afim de se realizar a prática do projeto de forma satisfatória como: baixo consumo, baixo custo, bom desempenho de processamento e capacidade de memória.

Diante dos requisitos estabelecidos, algumas alternativas disponíveis atualmente são elencadas em relação a microcontroladores. Entre algumas das opções de microcontroladores, estão os com a tecnologia chamada Atmel picoPower®, presente nos dispositivos da família AVR da Atmel Corp., a tecnologia de baixo consumo da *Texas Instruments*, presente nos dispositivos da família MSP430, além dos dispositivos de baixo consumo da família STM8 e STM32 da *STMicroelectronics* (marcados como *Ultra-Low-Power*)

Também tem a necessidade de se analisar a tecnologia para realizar o armazenamento de dados, podendo optar por uma memória flash, já presentes em algumas das placas de desenvolvimento dos microcontroladores listados ou realizar armazenamento em cartão de memória SD.

4.2 – ALTERNATIVAS PARA AS PLATAFORMAS DE *SOFTWARE*

Para desenvolvimento do projeto, é necessário também elencar as alternativas de *Integrated Development Environment* (IDE) disponíveis para as plataformas de hardware citadas. Para a família de microcontroladores da AVR, existe o Atmel® Studio, uma plataforma de desenvolvimento *freeware* e sem limitações.

Para a família MSP430, o Code Composer Studio™ é uma IDE *freeware* fornecida pela *Texas Instruments*. Já para a família STM8, é oferecido a IDE também gratuita ST Visual Develop e IAR Embedded Workbench ofertada gratuitamente por 30 dias. Por último a família STM32 tem as IDEs IAR Embedded Workbench for ARM

e DS-MDK ambas gratuitas por um período de avaliação e uma terceira IDE que é muito usada no mundo todo, por ser responsável também por um dos kit de desenvolvimento mais difundidos no mundo como a Arduino.

A construção de um dispositivo confiável e capaz de fornecer informações convenientes a vida do ser humano é esperada, com baixo consumo e facilidade em sua utilização.

5 - MATERIAIS E MÉTODOS

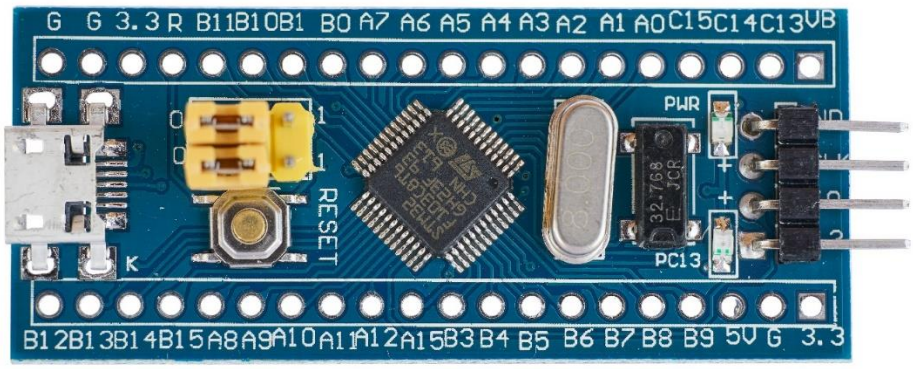
Os materiais e métodos utilizados para implementar o *holter*, serão apresentados neste capítulo.

5.1 – PLACA DE DESENVOLVIMENTO

Para a execução do sistema proposto, foi selecionado o kit de desenvolvimento conhecido por *BluePill*, com valor de R$7,00 se comprado na China. O kit de desenvolvimento possui um microcontrolador ARM Cortex-M3 de 32bits, o STM32F103C8T6, com *clock* de 72MHz, 64KB de memória *flash* e 20KB de SRAM. Como a proposta para o sistema é desenvolver um protótipo de baixo custo, nada melhor que a placa mais barata e mais popular da família STM32F103, que tem muito desenvolvimento tanto na plataforma da ARM, quanto pela IDE do Arduino, no qual é muito comparado.

O kit de desenvolvimento também conta com um RTC (*real-time clock*) com cristal de 32kHz, com um LED verde no pino PC13. Esta placa é muito comparada como sendo uma proposta melhor que Arduino Uno, que é um dos kits de prototipagem mais utilizados no mundo. Dentre as vantagens da BluePill, está a velocidade de processamento do microcontrolador e resolução do conversor analógico-digital maior. O kit de desenvolvimento é apresentado na Figura 11.

Figura 11: Visão superior do kit de desenvolvimento escolhido.

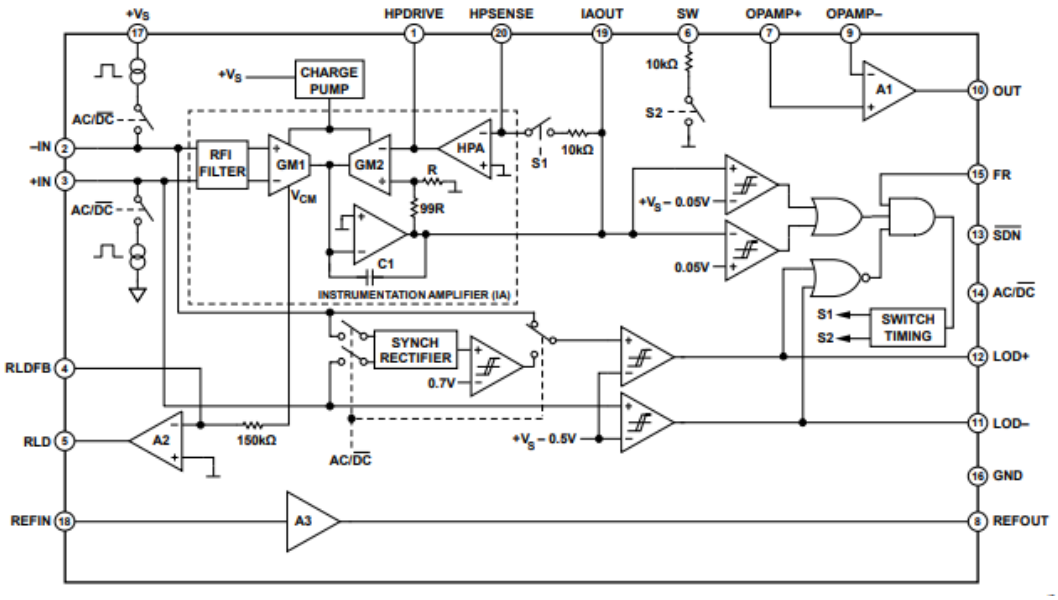


Fonte: (STM32, 2018).

5.2 – PLACA DE AQUISIÇÃO DE SINAIS DE ECG

O AD8232 é um *front-end* integrado para condicionamento de sinal de biopotenciais cardíacos para monitoramento da frequência cardíaca. Ele consiste de um amplificador de instrumentação especializado, um amplificador operacional (A1), um amplificador de acionamento para a perna direita (A2) e um buffer de referência de meio da frase (A3). Além disso, o AD8232 inclui um circuito de detecção de derivações e um circuito de restauração rápida automática que retorna o sinal logo após a reconexão dos terminais (ANALOG DEVICES, 2018). O esquemático simplificado é apresentado na Figura 12.

Figura 12: Esquemático AD8232.

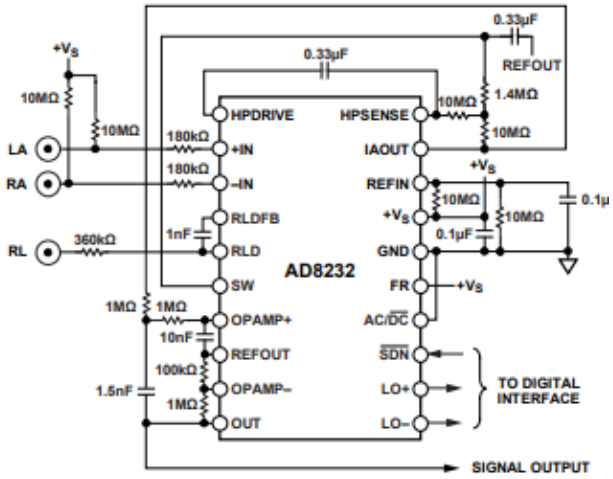


Fonte: (ANALOG DEVICES, 2018).

O AD8232 contém um amplificador de instrumentação especializado que amplifica o sinal de ECG enquanto rejeita o potencial de meia-célula do eletrodo no mesmo estágio. Isso é possível com uma arquitetura de feedback de corrente indireta, que reduz o tamanho e a potência em comparação com as implementações tradicionais (ANALOG DEVICES, 2018).

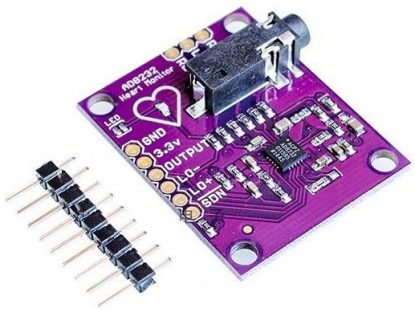
A placa escolhida foi projetada para monitorar a forma da forma de onda do ECG na configuração de monitor cardíaco. Ele assume que o paciente permanece relativamente imóvel durante a medição e, portanto, os artefatos de movimento são um problema menor. É apresentado o circuito da configuração citada na Figura 13 e na Figura 14 a placa de aquisição.

Figura 13: Circuito da configuração de Monitor Cardíaco.



Fonte: (ANALOG DEVICES, 2018).

Figura 14: Placa de aquisição de sinais.

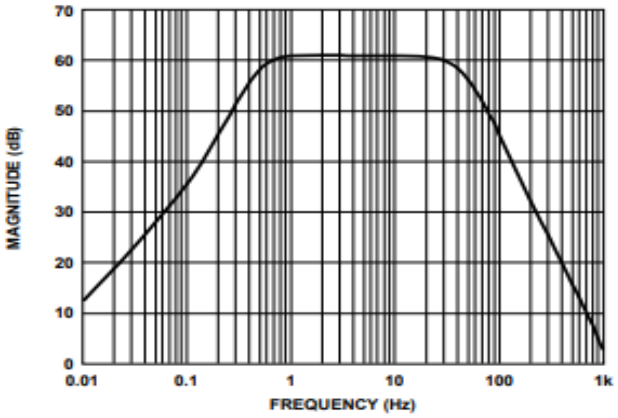


Fonte: (ALIEXPRESS, 2018).

Assim, para obter uma forma de onda com o mínimo de distorção possível, o AD8232 foi configurado com um FPB de 0.5 Hz com dois polos, seguido de um FPA também com dois polos e frequência de corte em 40 Hz. Um terceiro eletrodo é acionado para uma ótima rejeição de modo comum, que no caso é o eletrodo da perna direita. Além da filtragem de 40 Hz, o estágio do amplificador operacional é

configurado para um ganho de 11, resultando em um ganho total do sistema de 1100 (ANALOG DEVICES, 2018). O gráfico da resposta em frequência da configuração de monitor cardíaco e apresentado na Figura 15.

Figura 15: Resposta em frequência do monitor cardíaco.



Fonte: (ANALOG DEVICES, 2018).

Mas por se tratar de um exame, que se realizaria em um ser humano, optou-se por utilizar um simulador de ECG, para que possa atestar o funcionamento do protótipo em questão. O simulador foi desenvolvido utilizando o mesmo kit de desenvolvimento, o STM32 ou BluePill, e um conversor digital-analógico (DAC) MCP4725 conectados via I2C e saída conectada ao ADC do protótipo sem o módulo AD8232. Este conversor é muito indicado para projetos de instrumentos musicais, gerador de sinais e outras aplicações.

O sinal utilizado pelo simulador para realização do projeto, foi retirado do MIT-BIH *Arrhythmia Database*, como uma amostra de dez minutos a 12 bits de resolução e encontrado na fonte com *record number* 200 (MIT-BIH Arrhythmia Database, 2018).

5.3 – FONTE DE ALIMENTAÇÃO

O kit de desenvolvimento STM32 é alimentado, com tensão de 5V em corrente contínua. Sendo o sistema proposto um dispositivo portátil, verificou-se a necessidade de alimentar o kit de desenvolvimento através de bateria.

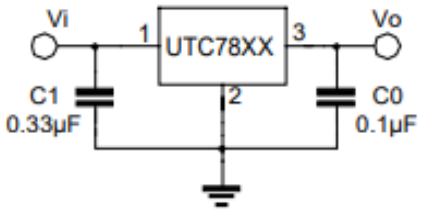
Os microcontroladores são responsáveis por fornecer alimentação aos dispositivos que estão conectados entre si, fornecendo uma diferença de potencial de 3.3V através de seus reguladores de tensão. Assim foi realizada por não

sobrecarregarem seus respectivos reguladores, e não se tratar de um grande número de dispositivos interligados.

Mas, como fonte de alimentação dos microcontroladores, por simplicidade, adotou-se baterias de lítio de 3.7V, sendo duas ligadas em série, fornecendo 7.4V, que podem ser facilmente encontrada no mercado e serem eficientes, que apresenta na folha de informações ter 1850mAh (DYNAPACK). A fonte de tensão conta com um regulador de tensão UTC7805 e capacitores, como especificado na folha de informações do componente. A Figura 16 e a Figura 17 apresentam o esquemático da fonte de tensão e o esquemático de alimentação entre todos os componentes, respectivamente.

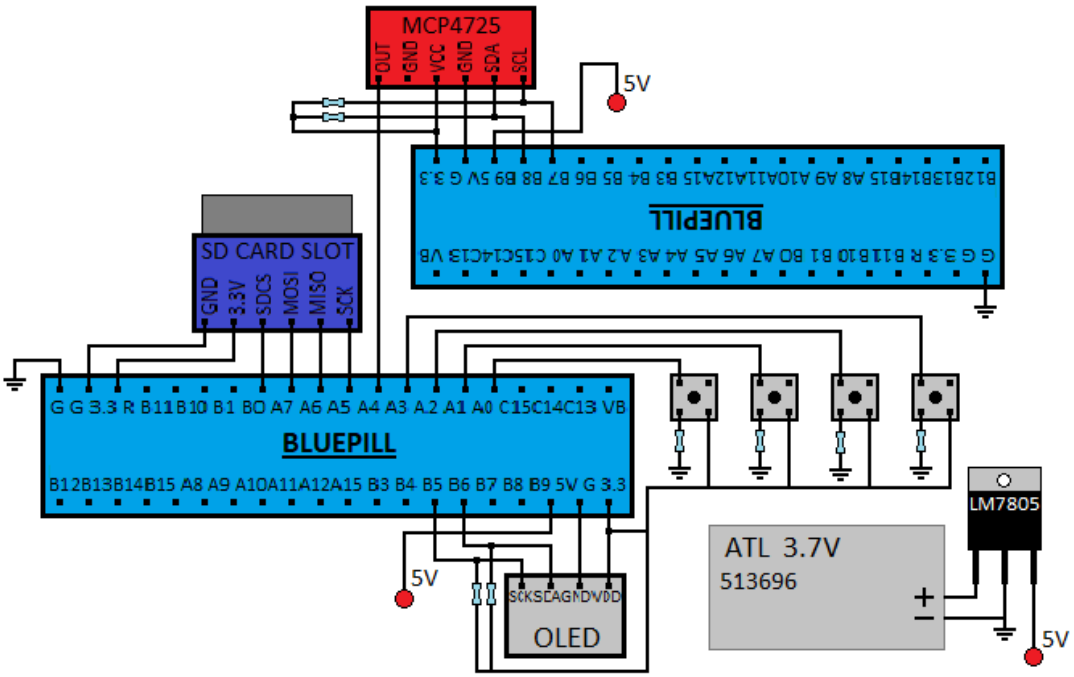
.

Figura 16: Esquemático da simples fonte de tensão.



Fonte: (YOUW ANG ELECTRONICS CO).

Figura 17: Esquemático do sistema.

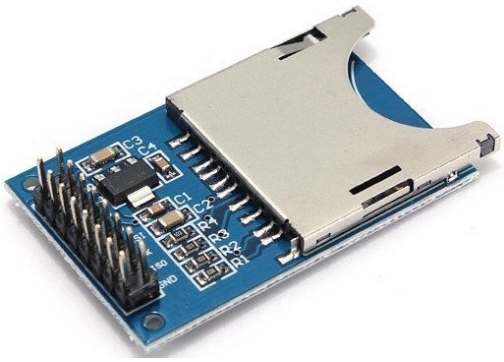


Fonte: Autoria própria

5.4 – PLACA DO CARTÃO DE MEMÓRIA

Com o intuito de armazenar as informações geradas pela aplicação, tais como o sinal de ECG gerado e momentos de possível desconforto por parte do paciente durante o exame realizado, foi selecionado uma placa de cartão de memória que é comumente utilizada. Trata-se da mesma utilizada em muitos kits de prototipagem, pelo baixo custo e facilidade de uso e que se comunica com o microcontrolador através do protocolo de comunicação SPI. A Figura 18 apresenta o dispositivo de armazenamento dos dados gerados na aplicação.

Figura 18: Módulo cartão de memória.



Fonte: (BAÚ DA ELETRÔNICA, 2018).

5.5 – INTERFACE HOMEM MÁQUINA

Como parte da interface do dispositivo com o humano, algumas chaves tácteis ou push-buttons, que são bastante empregados em prototipagem de projetos, foram utilizados na para acessar os menus e configurar o mesmo. Além do monitor escolhido para a aplicação, com baixo consumo de energia e uma alternativa de baixo custo, OLED SSD1306 de 0.96 polegadas com resolução de 128x64 pixels, que se comunica com o microcontrolador através do protocolo de comunicação I2C.

Para auxiliar no desenvolvimento da aplicação, foi utilizado também uma protoboard, visto que se trata da integração de todos estes componentes já citados.

5.6 - TIMER E CONVERSOR ANALÓGICO DIGITAL

Como já visto, para representar os sinais convertidos com fidelidade devem ser obedecidas duas propriedades, a taxa de Nyquist e a quantização.

Assim, a taxa de amostragem deve ser maior que o dobro da frequência do sinal amostrado. Para que isso ocorra, alguns cálculos foram realizados afim de configurar o timer para acionar o conversor analógico digital e coletar a amostra do sinal.

O *timer* tem 16 bits de contagem, chegando ao estouro em 65535, e a frequência do processador utilizado na aplicação é de 72 MHz (F_{CPU}). Amostrando o sinal em 240 Hz (F_S), é necessário escolher o valor a ser passado ao registrador do *prescaler* (PSC), para que o registrador do estouro (ARR) seja preenchido, a partir da Equação 2.

$$PSC > \frac{F_{CPU}}{F_S * ARR}. \qquad \text{Equação 2.}$$

$$PSC > {}^{72.10^6}/_{240 * 2^{16}}$$

$$PSC > 4{,}57.$$

Com base no manual de referência do microcontrolador, obtêm-se o *clock* efetivo (F_{CNT}) do timer a partir da Equação 3.

$$F_{CNT} = \frac{F_{CPU}}{PSC+1}. \qquad \text{Equação 3.}$$

$$PSC = 7, \text{ escolhido.}$$

$$F_{CNT} = 9\ MHz.$$

Com os valores de *clock* efetivo e de prescaler definidos, é possível estabelecer o valor do registrador de *auto-reload* do *timer*, através da Equação 4 a seguir.

$$ARR = \frac{F_{CNT}}{F_S}. \qquad \text{Equação 4.}$$

$$ARR = {}^{9.10^6}/_{240}$$

$$ARR = 37500.$$

O registrador de controle do *timer* foi configurado no modo de evento de atualização, para gerar interrupção assim que o estouro é alcançado, e o ADC ser acionado.

Continuando com as configurações, para que a coleta das amostras represente o sinal analógico com fidelidade, no registrador de controle do ADC foram setados o acionamento por evento externo, no caso o *timer* realiza esta função, indicar qual evento realizará o acionamento e também a tensão de referência para as conversões (STMICROELETRONICS). Com a tensão de referência (V_{REF}) e resolução do ADC (RES_{ADC}) de 12 bits , que fornece uma boa conversão e um baixo erro de quantização, a sensibilidade da conversão (S_{ADC}) é apresentada a seguir com a Equação 5.

$$S_{ADC} = \frac{V_{REF}}{RES_{ADC}}. \qquad \text{Equação 5.}$$

$$S_{ADC} = {}^{3.3}/_{2^{12}}$$

$$S_{ADC} = 805{,}66\mu V.$$

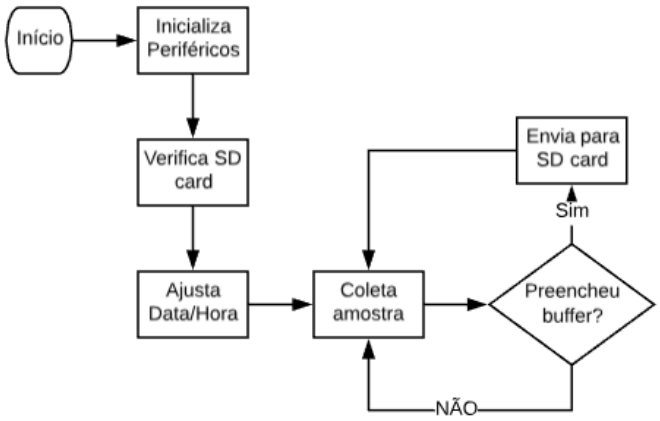
Com os parâmetros estabelecidos, microcontrolador configurado é esperado que se realize a conversão do sinal de natureza contínua para um sinal discretizado, com fidelidade.

5.7 – FLUXOGRAMA DA ROTINA PRINCIPAL

Este capítulo mostra como foi executado o desenvolvimento do firmware da aplicação, as ferramentas utilizadas e as etapas a serem seguidas para executar as tarefas.

O diagrama de fluxo da rotina principal é apresentado na Figura 19 e em seguida suas devidas partes.

Figura 19: Fluxograma base para as atividades.



Fonte: Autoria Própria.

Todas as tarefas a serem realizadas pelo microcontrolador foram desenvolvidas através da Arduino IDE, que facilita bastante a integração de dispositivos, oferecendo suporte a diversas plataformas e bastante difundido no mundo todo.

Sendo assim, os programas baseados em *Sketch,* são divididos em duas partes, o setup e o loop. Durante o Setup são inicializados e configurados os periféricos a serem utilizados pela aplicação e caso algo não suceda como planejado, permanece na configuração.

Durante o loop, a aplicação acontece sendo dividida em outras duas partes, sendo uma de configuração de Data e Hora do sistema e a outra de gravação das amostras de ECG.

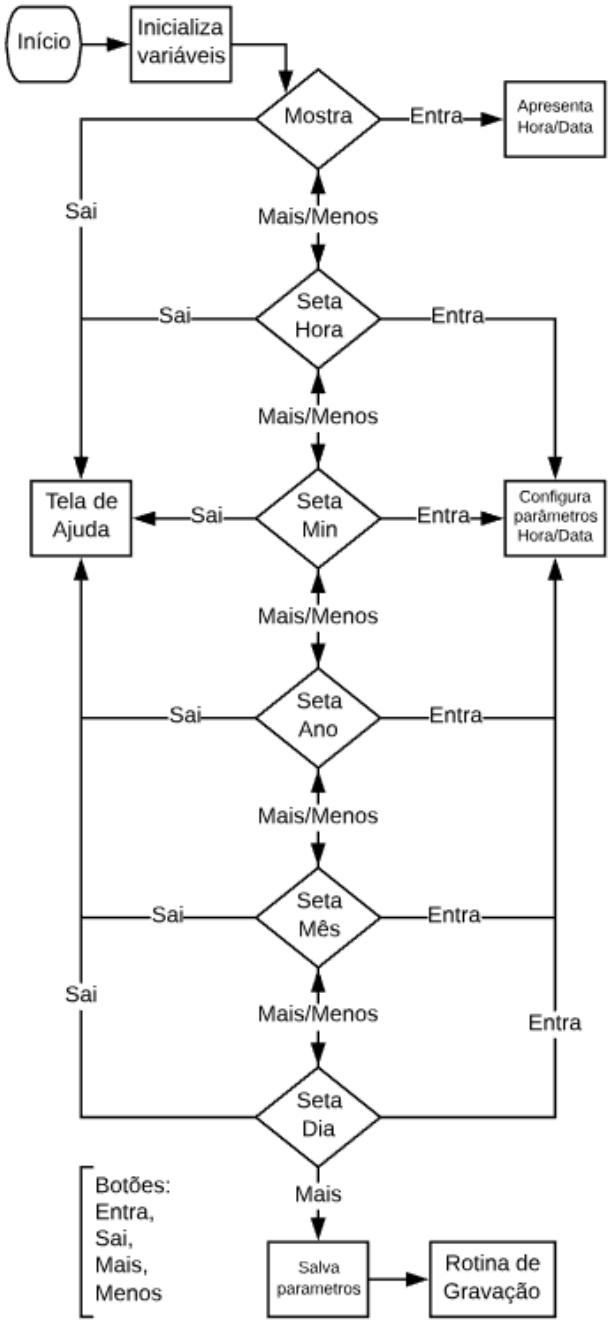
Foi necessário incluir um URL adicional no gerenciador de placas da IDE, para que a mesma possa fornecer informações da placa utilizada, a STM32F103C8T6 e assim prosseguir mais facilmente. Também foram incluídas algumas bibliotecas, que são bastante utilizadas nesse ambiente de desenvolvimento voltadas a esta placa e baixadas pelo repositório GitHub.

Foram adicionados ao programa as bibliotecas referentes ao relógio de tempo real (RTClock.h), ao cartão de memória (SdFat.h e SPI.h), ao monitor OLED (Wire.h, Adafruit_SSD1306_STM32.h e Adafruit_GFX_AS.h) e também a biblioteca para configurar o ADC e timer, fazendo possível coletar as amostras com frequência de amostragem desejada.

5.8 – FLUXOGRAMA DA CONFIGURAÇÃO DE HORA/DATA

Este capítulo mostra como funciona a etapa de configuração de hora e data através dos menus gerados para aplicação, com acesso aos mesmos através dos botões para IHM. A seguir é apresentado o diagrama de fluxo do firmware desenvolvido para tais configurações na Figura 20.

Figura 20: Fluxograma da rotina de configuração de Hora/Data.



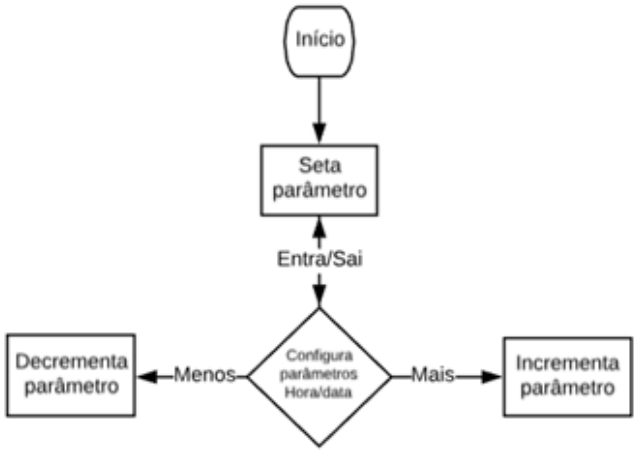
Fonte: Autoria própria.

Dentre as etapas citadas no diagrama de fluxo, a ‘Tela de Ajuda’ é apresentada durante três segundos, e retorna ao ponto onde estava, com o propósito

de informar qual a função de cada um dos botões de IHM. Assim, já é possível navegar pela aplicação com os botões ‘Menos’, ‘Mais’, ‘Entra’ e ‘Sai’.

Já a etapa ‘Mostra’, quando apertado o botão ‘Entra’, é apresentado hora e data gravadas no sistema e retorna ao ponto onde estava ao pressionar o Botão ‘Sai’. A etapa de ‘Salva parâmetros’, é a última etapa da primeira parte da aplicação, que vai salvar os valores selecionados na etapa ‘Configura parâmetros Hora/Data’ que é apresentado no fluxograma de processos na Figura 21.

Figura 21: Fluxograma de atividade.

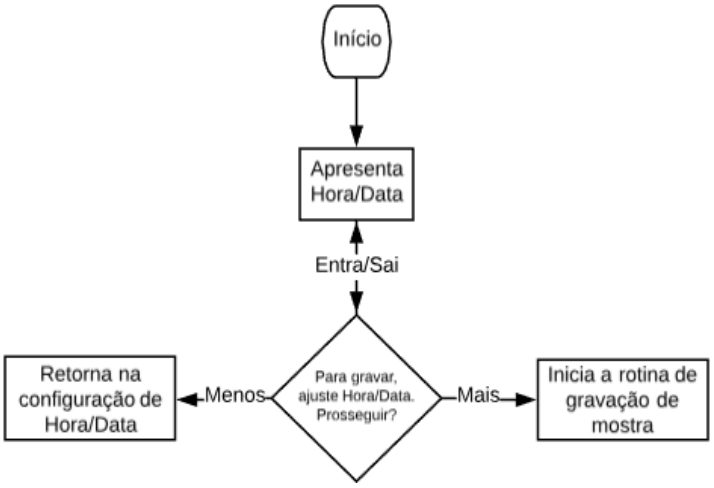


Fonte: Autoria própria.

5.9 – FLUXOGRAMA DA ROTINA DE GRAVAÇÃO

Este capítulo apresenta a segunda parte da aplicação, da qual trata da gravação das amostras recebidas do ADC em arquivos no cartão SD. Após configurar hora e data e confirmar a ação, se faz possível iniciar a gravação das amostras coletadas através do ADC e o fluxograma do firmware desenvolvido para realizar a tarefa é apresentado na Figura 22.

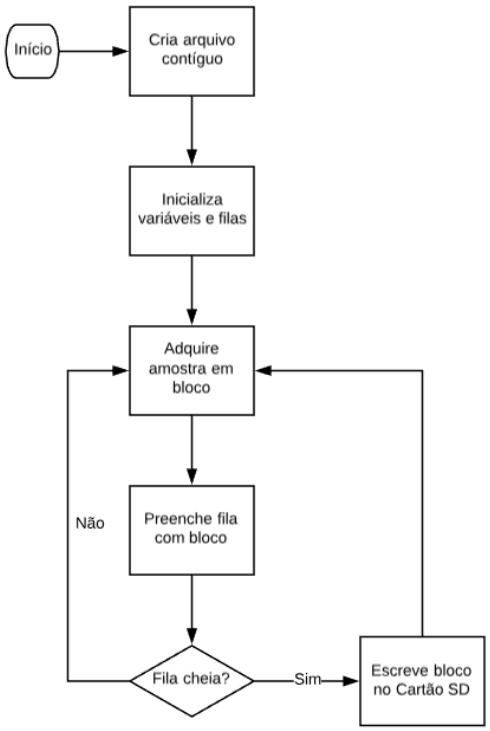
Figura 22: Fluxograma para início das gravações.



Fonte: Autoria Própria.

A gravação das amostras é realizada com auxílio de filas e com preenchimento de arquivo contíguo, que são alocados de modo sequencial e não fragmentados, tornando o preenchimento otimizado e diminuindo a latência de escrita. É apresentado na Figura 23, o diagrama de fluxo da gravação das amostras.

Figura 23: Fluxograma da rotina de gravação.



Fonte: Autoria própria.

Ao final da gravação das amostras em arquivo do tipo .BIN, um arquivo do tipo .CSV é gerado, contendo as amostras dos dois canais separados por vírgula, provenientes do arquivo .BIN gerado. Também é gerado um arquivo do tipo .TXT, do qual contém informações de hora e data de momentos gravados durante a aquisição dos dados no arquivo .BIN.

5.10 – SOFTWARE PARA LEITURA DAS AMOSTRAS

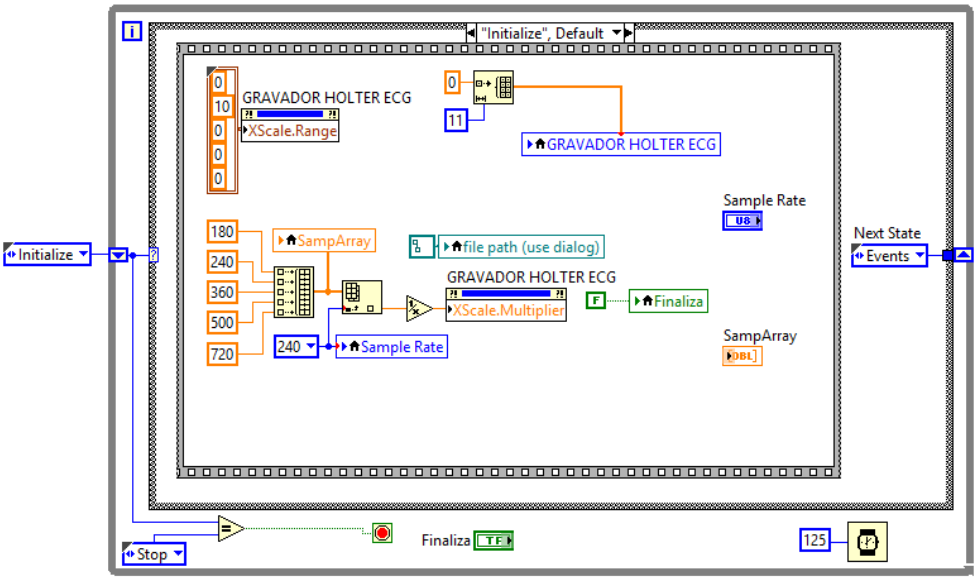
Este é um software desenvolvido para a aplicação, através do LabVIEW capaz de ler o arquivo do tipo .BIN gerado pela aplicação e apresentar as formas de ondas descritas pelas amostras coletadas. Basta ter instalado no computador que realizará a leitura o *runtime engine* do LabVIEW, e assim executar o *Virtual Instrument* (VI) denominado Holter.

O desenvolvimento é feito de acordo com modelo de fluxo de dados, através de programação gráfica, que permite aliar rapidamente tarefas como aquisição de

dados, análise e operações lógicas, e ainda compreender como os dados estão sendo modificados (NATIONAL INSTRUMENTS CORPORATION).

A seguir serão apresentados os diagramas desenvolvidos para execução do VI. Este VI, foi projetado como uma máquina de estados onde o primeiro estado é apresentado o diagrama para inicialização de todos os componentes utilizados na aplicação, na Figura 24.

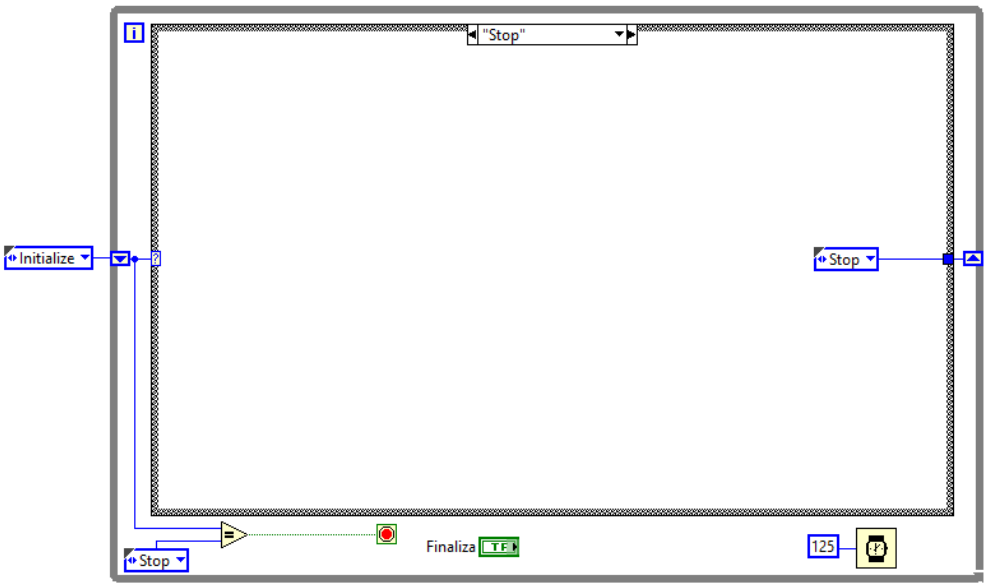
Figura 24: Estado de inicialização.



Fonte: Autoria própria.

O próximo diagrama apresentado na Figura 25, é o estado de finalização do VI, quando se pressiona a tecla ‘Finaliza’.

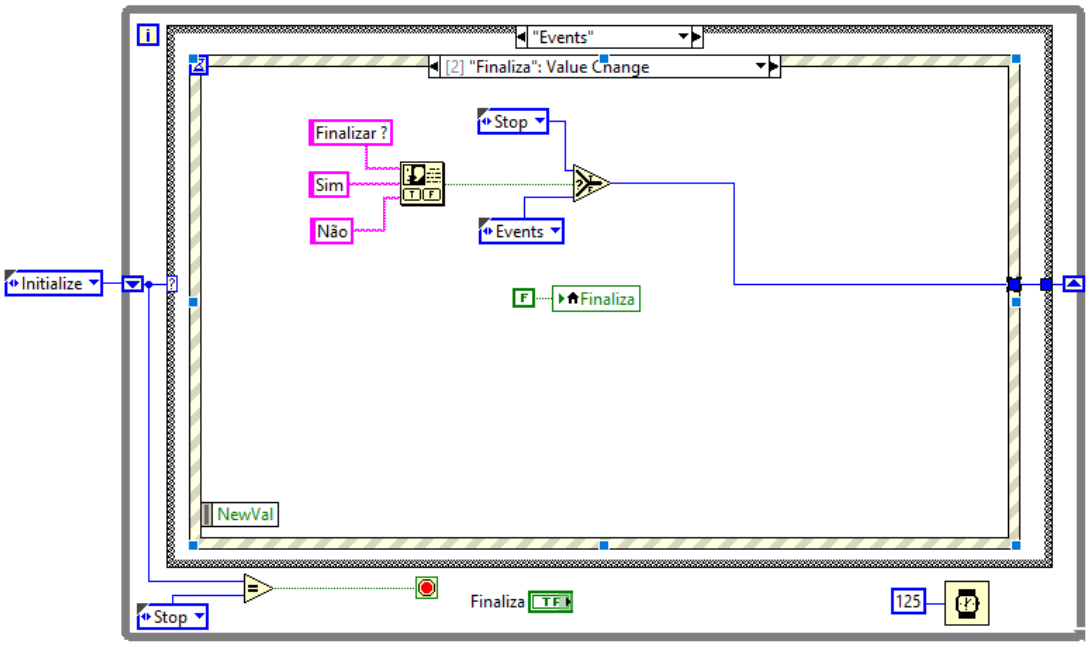
Figura 25: Estado de parada.



Fonte: Autoria própria.

O próximo estado da máquina de estados é o estado ‘Events’, o qual monitora através de uma estrutura de eventos os controles do painel frontal, a saber, o controle de abertura de arquivos de ECG, o controle de seleção de taxa de amostragem e o controle ‘Finaliza’. Na Figura 26 é apresentado o subdiagrama do evento para confirmação da ação associado ao controle ‘Finaliza’.

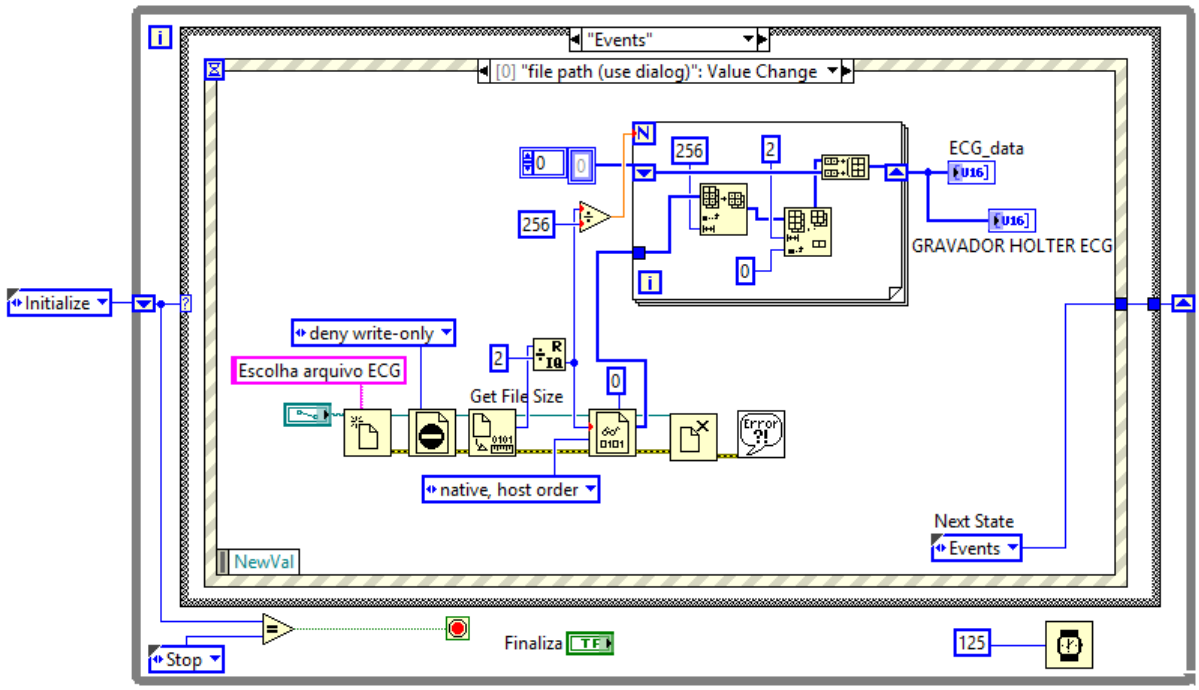
Figura 26: Evento para finalização da aplicação.



Fonte: Autoria própria.

Outro subdiagrama desenvolvido para aplicação, é referente ao evento de abertura de arquivo de dados de ECG para leitura das amostras geradas pelo protótipo *Holter*. Na figura 27 é apresentado o subdiagrama responsável por executar a ação citada, onde o bloco ‘for’ serve para eliminar os preâmbulos no início de cada bloco de dados de 256 amostras.

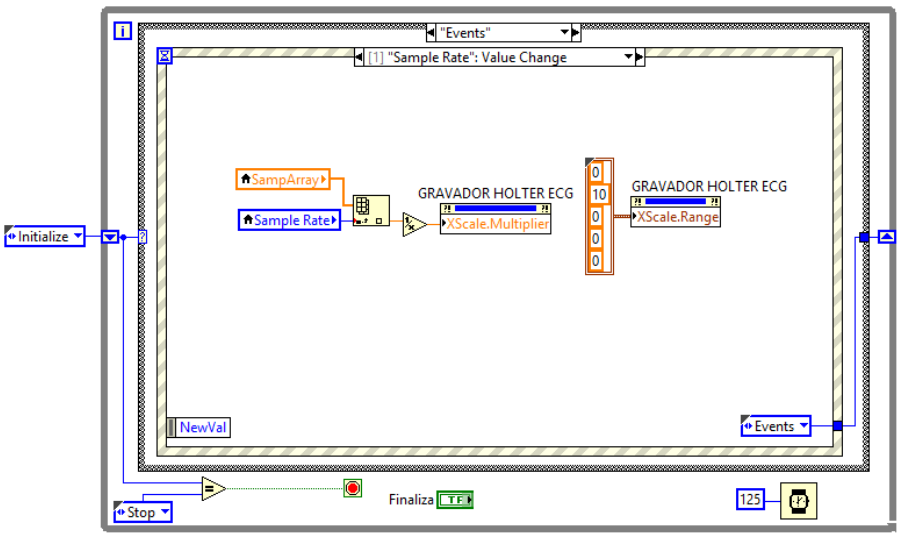
Figura 27: Evento para abrir arquivo.



Fonte: Autoria própria.

O próximo subdiagrama apresentado, é responsável pela seleção da taxa de amostragem do arquivo de dados a ser aberto, sendo as seguintes taxas possíveis: 180, 240, 360, 500, 720 Hz. Na Figura 28 é apresentado o desenvolvimento do referido evento.

Figura 28: Evento de seleção da taxa de amostragem.



Fonte: Autoria própria.

5.11 - O PROTÓTIPO

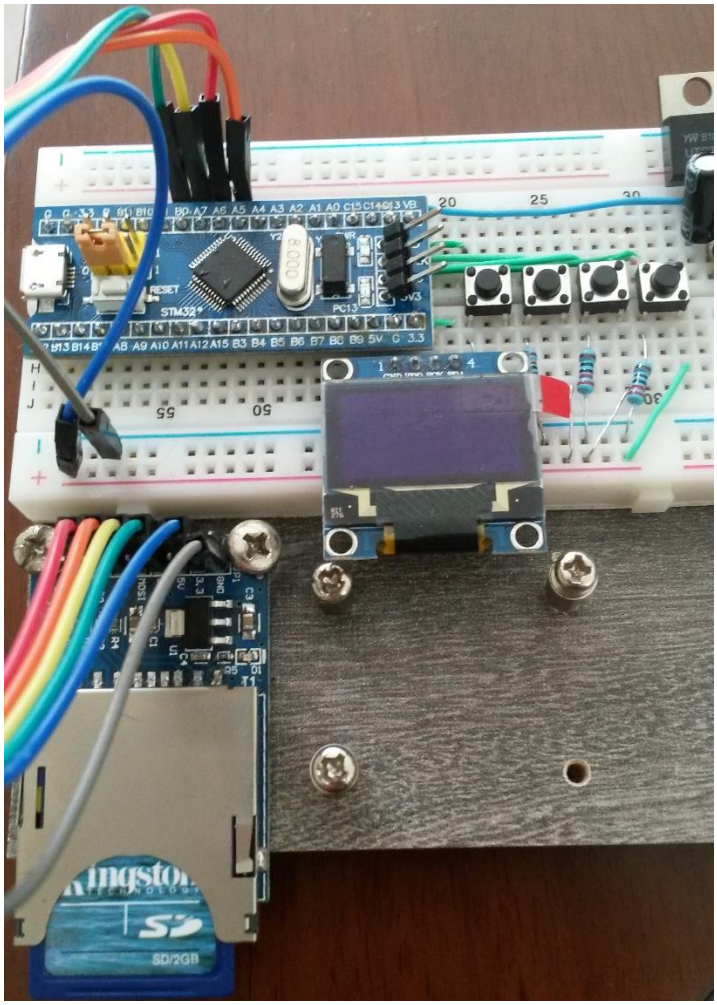
O desenvolvimento do sistema proposto se deu com auxílio de uma protoboard, responsável pela conexão dos componentes utilizados na aplicação, que é o mais comum a ser realizado quando se pretende construir o primeiro protótipo.

Foram utilizadas algumas chaves-tácteis ou *push-buttons*, para que possa acessar os menus desenvolvidos para ajuste de hora e iniciar a gravação das amostras. Os pinos utilizados para tais botões no microcontrolador STM32F103C8T6 foram os A0, A1, A2 e A3, conectados com *pull-up*, sendo responsáveis por tais ações descritas.

Outro dispositivo que faz parte da interface homem-máquina é o monitor OLED, que com o auxílio do mesmo se faz possível apresentar as horas a serem ajustadas pelo usuário para iniciar a gravação das amostras e também gravar a hora em que houver um desconforto por parte do paciente. O monitor é conectado pelos pinos B6 e B7 do microcontrolador, sendo SCL e SDA respectivamente da conexão via I2C. Os outros pinos do monitor de alimentação foram conectados a 3.3V e GND.

O dispositivo de armazenamento é o responsável pelo armazenamento das amostras coletadas pelo ADC através do pino A4, conectado aos pinos A5, A6, A7 e B0, sendo SCK, MISO, MOSI e NSS, respectivamente. Os pinos de alimentação também conectados a 3.3V e GND do microcontrolador. A Figura 29 apresenta o protótipo como um todo.

Figura 29: Protótipo de aquisição de sinais.



Fonte: Autoria Própria.

5.12 - O SIMULADOR

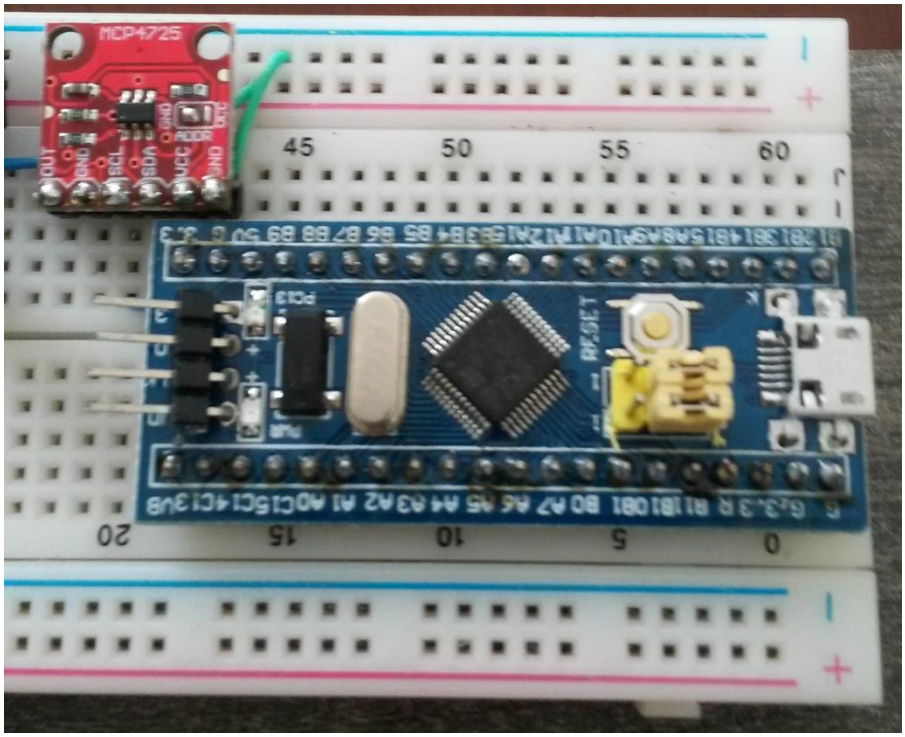
Em virtude de se tratar de biopotenciais elétricos, optou-se pelo uso de um simulador de sinais, ao invés do circuito AD8232, que também foi elencado como alternativa.

Foi utilizado outro microcontrolador STM32F103C8T6 para o protótipo do simulador, compartilhando a mesma protoboard do sistema de aquisição de sinais, facilitando assim o desenvolvimento de todo o trabalho, pois compartilham também a fonte de alimentação.

Além do microcontrolador, faz parte também do simulador, o DAC, que vai conectado aos pinos B8 e B9, sendo SCL e SDA respectivamente e fonte de

alimentação em 3.3V. O dispositivo do DAC se comunica com o microcontrolador através de protocolo I2C, e utiliza sinal do banco de dados MIT-BIH, que é enviado ao pino A4 do protótipo de aquisição pela saída do DAC. Na Figura 30 é apresentado o protótipo em questão.

Figura 30: Simulador de ECG.



Fonte: Autoria Própria.

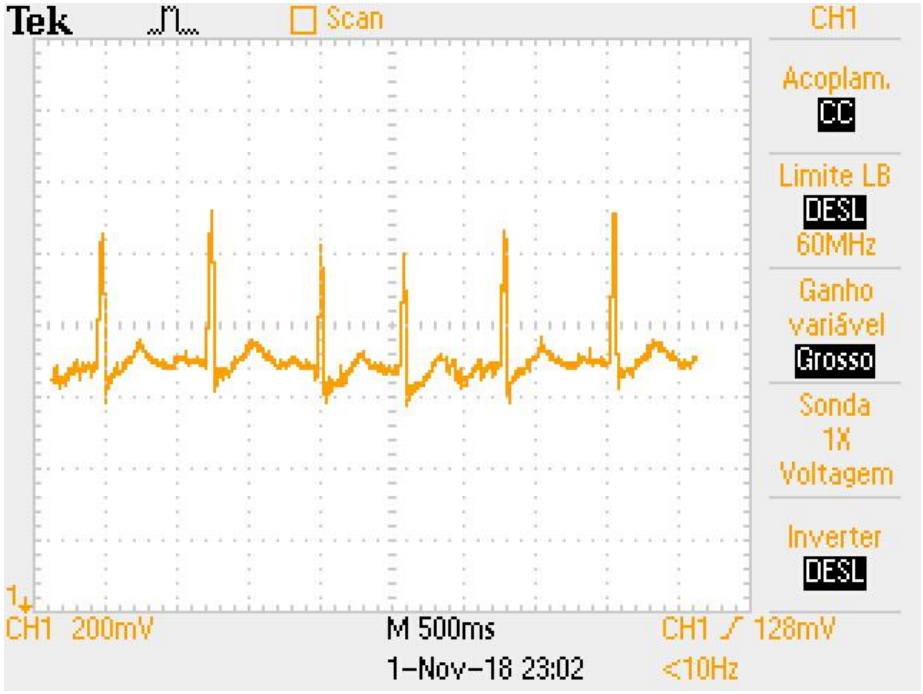
6 – RESULTADOS

Este capítulo abordará os resultados obtidos com o desenvolvimento deste trabalho, o funcionamento do protótipo e a viabilidade do mesmo como proposta de um novo produto.

6.1 – DO SINAL DO SIMULADOR

O protótipo do simulador funcionou como o esperado e o sinal gerado através do mesmo foi analisado com auxílio de um osciloscópio e apresenta uma boa amplitude de sinal, como é apresentado na Figura 31.

Figura 31: Sinal de saída do simulador.



Fonte: Autoria própria.

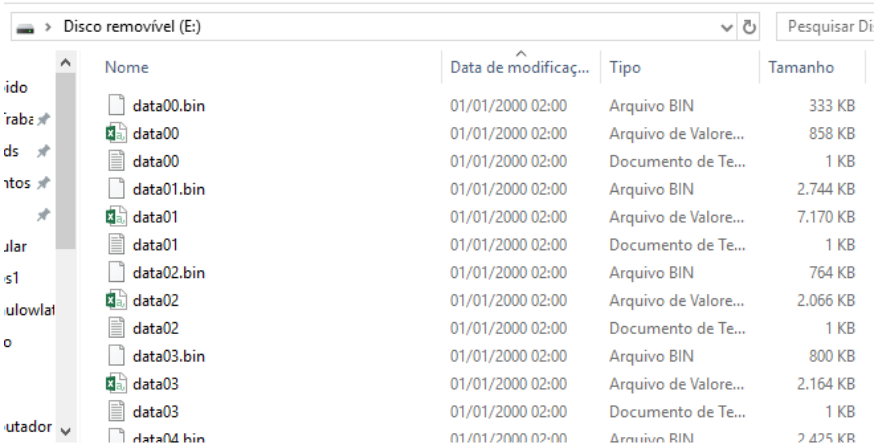
6.2 - DO ARQUIVO DE AMOSTRAS

O armazenamento das amostras coletadas pela aplicação é enviado para o cartão SD, alocada em arquivo contíguo criado e gerenciado pela mesma. O arquivo gerado é do tipo BIN (Binário) e o mesmo é convertido ao final da gravação para um arquivo do tipo CSV (valores separados por vírgula), ambos levam o mesmo nome, DATA e mais dois algarismos começados em 00, de acordo com os arquivos já existentes no cartão de memória. Com esse tipo de arquivo, os resultados coletados podêm ser facilmente utilizados em softwares capazes de gerar gráficos para análise das amostras coletadas.

6.3 - DO ARQUIVO DE MOMENTOS COLETADOS

Como proposta da aplicação, após ajuste de hora e data do RTC, requeridos para executar o início da coleta de amostras, optou-se por deixar um dos botões para gravar um momento de possível desconforto durante o exame. Tal tarefa só é possível de ser executada durante a gravação das amostras, armazenadas em arquivo do tipo TXT, levando o mesmo dos arquivos gerados para coletar as amostras do sinal de ECG. A Figura 32 apresenta os arquivos armazenados no cartão SD. e o arquivo com amostras de tempo coletadas.

Figura 32: Arquivos gerados no cartão de memória.



Fonte: Autoria própria

Mas não obtive sucesso ao tentar realizar a gravação dos momentos de desconforto, pois a interrupção durante a gravação das amostras, trunca o arquivo.

6.4 - DA FONTE DE ALIMENTAÇÃO DOS MICROCONTROLADORES

Os microcontroladores são responsáveis por fornecer alimentação aos dispositivos que estão conectados entre si, fornecendo uma diferença de potencial de 3.3V. Assim foi realizada por não sobrecarregarem seus respectivos reguladores, por não se tratar de um grande número de dispositivos interligados.

O consumo dos protótipos utilizados na aplicação foram mensurados e são apresentados, na Figura 33 e na Figura 34, sendo os valores de consumo do simulador e do *holter*, respectivamente.

Figura 33: Consumo de corrente do simulador.

Fonte: Autoria própria.

Figura 34: Consumo de corrente do holter

Fonte: Autoria própria.

A medição foi realizada com auxílio de um multímetro, em escala de 200m e cabos de prova conectados na entrada de mA. O consumo de energia do protótipo *holter* foi calculado e é apresentado na equação 6, para que seja possível mensurar o a quantidade de energia deve ser fornecida ao sistema para se manter ligado por 24 horas.

$$C_w = V * I * T. \qquad \text{Equação 6.}$$
$$C_w = 5 * 54m * 24$$
$$C_w = 6.84\ Wh$$

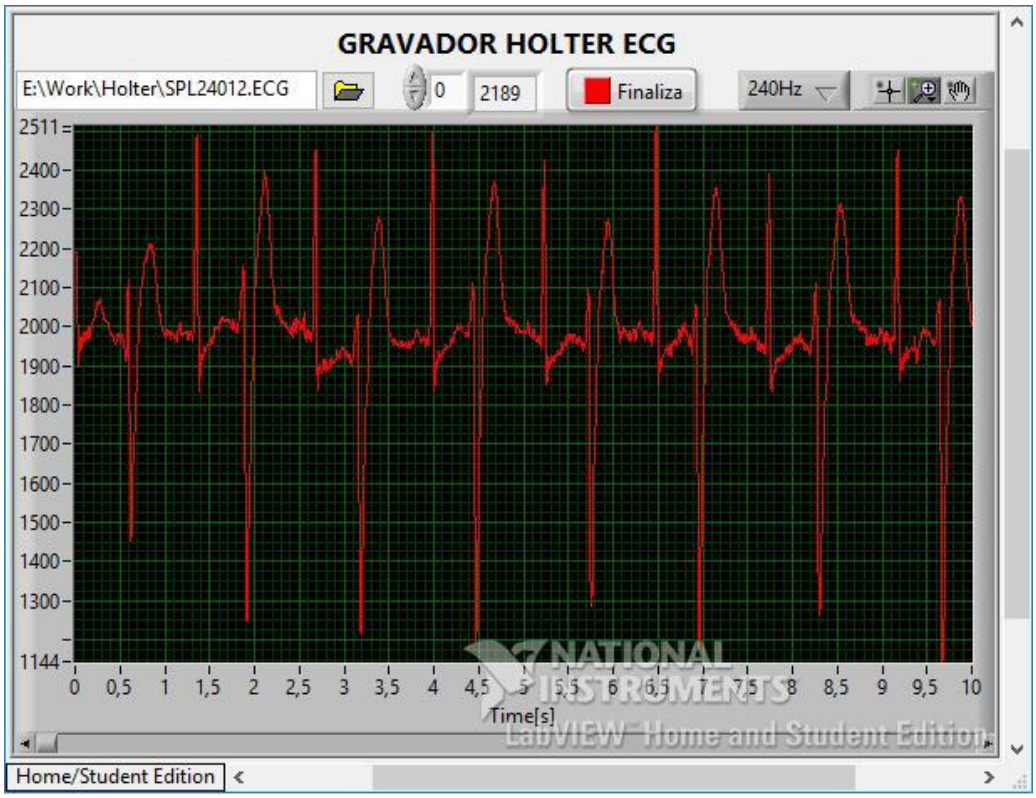
Sendo as baterias de 3.7V utilizadas em série, que fornecem 1850mAh cada, tornando um conjunto capaz de se manter durante um ciclo inteiro.

6.5 - DO *VIRTUAL INSTRUMENT*

Para leitura das amostras coletadas pelo protótipo de aquisição de sinais com uso de cartão de memória, o VI Holter, software desenvolvido para a aplicação, foi utilizado.

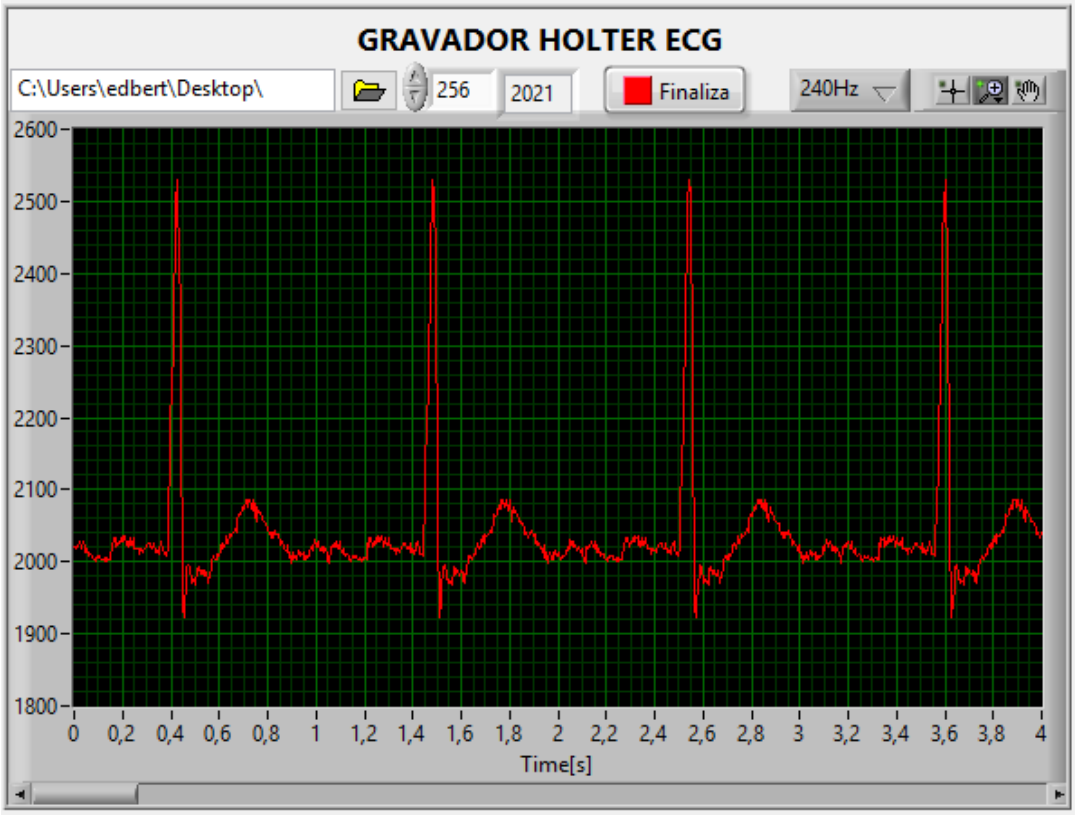
A figura 35 e a figura 36 apresentam o layout do software desenvolvido com sinal de arritmia, e a forma de onda resultante das amostras coletadas pelo *Holter*.

Figura 35: Layout do software desenvolvido para leitura das amostras.



Fonte: Autoria Própria.

Figura 36: Sinal com amostras coletadas pelo protótipo.



Fonte: Autoria própria.

6.6 – DA APLICAÇÃO

A interação com o dispositivo se dá com auxílio das ferramentas de interface homem-máquina, para que possa ajustar os parâmetros antes de iniciar a gravação do sinal de ECG. Sem que estes parâmetros sejam ajustados, ou que não exista cartão de memória no *slot*, a aplicação não se inicia. A Figura 37 apresenta a mensagem sinalizada.

Figura 37: Mensagem por falta de cartão SD.

Fonte: Autoria própria

Após a inicialização, são apresentadas as opções para ajuste de Hora e Data e mais uma opção, para visualizar os ajustes realizados. Além do monitor OLED, mais quatro chaves-tácteis são utilizados para realizar ações descritas, sendo apresentadas na Figura 38 as funções de cada botão, para acesso e ajuste aos menus existentes.

Figura 38: Tela de Ajuda.

Fonte: Autoria própria.

Algumas variáveis foram atribuídas ao sistema para que seja possível o acesso aos menus, criando condições das quais selecionam cada etapa da aplicação, realizando com o incremento ou decremento das variáveis para entrar ou sair dos menus existentes. A aplicação é dividida em duas partes, sendo a primeira delas com seis menus, sendo um para mostrar hora e data e os outros cinco para configurar, hora, minuto, dia, mês e ano. A Figura 40 apresenta o menu MOSTRA, com hora e data configurados.

Figura 39: Tela do menu MOSTRA.

Fonte: Autoria própria.

A segunda parte da aplicação, apresenta também hora e data configurados, e caso estejam corretos, inicia-se a gravação das amostras pressionando o botão ENTRA, caso contrário a aplicação seta a variável flagrtcok em 0 pressionando o botão SAI, e inicia a primeira parte da aplicação novamente. Durante a segunda parte da aplicação, apenas os botões ENTRA e SAI têm funções, sendo o primeiro para gravar os momentos em que se possa ter um desconforto para futuras análises e o segundo para finalizar a gravação das amostras coletadas a umas taxa de 240 Hz em dois canais do ADC. A seguir a Figura 40 e a Figura 41 apresentam as telas para iniciar a gravação e a tela durante a gravação.

Figura 40: Tela para início das gravações

Fonte: Autoria própria.

Figura 41:Tela durante gravação

Fonte: Autoria própria.

7 – CONCLUSÕES

Visto o grande número de pessoas atingidas no mundo todo por doenças cardiovasculares, é de grande importância o desenvolvimento de dispositivos que auxiliem nos diagnósticos dessas doenças o mais rápido possível, e fornecendo acesso as pessoas através de dispositivos de baixo custo.

O dispositivo desenvolvido é de suma importância para diagnóstico de arritmias, que tem caráter transitório, conseguindo fazer longas gravações para posterior análise.

Em se tratando de um dispositivo portátil, de longa duração de bateria, a escolha de baterias de lítio, foi suficiente para um período de 24 horas de gravações e ainda, fornecendo alimentação também para o simulador, que tem consumo superior ao dispositivo de entrada, ou circuito de aquisição de sinais de ECG.

A escolha pelo kit de desenvolvimento da STM32 e a possibilidade de se trabalhar com a IDE Arduino, tornou o desenvolvimento do sistema em questão, uma boa base para futuros projetos mais elaborados, considerando o cronograma reduzido para realização do mesmo.

Todos os dispositivos utilizados são facilmente encontrados no mercado, de fácil integração entre si e que tornaram um projeto de baixo custo, com o valor dos dispositivos apresentados na Tabela 3.

Tabela 3: Valor dos dispositivos

CUSTOS DO PROJETO			
Item	Valor(R$)	Quantidade	Subtotal(R$)
Kit BluePill	6,45	2	12,90
AD8232	13,43	1	13,43
Monitor OLED	6,65	1	6,65
Módulo Cartão SD	2,10	1	2,10
Módulo DAC	2,83	1	2,83
Baterias 3.7v	31,87	2	63,74
Protoboard	4,82	1	4,82
		Total	106,47

Fonte: Autoria própria.

7.1 – TRABALHOS FUTUROS

A comercialização de dispositivos da área médica, podem ter boa rentabilidade, mas demandam tempo e muito trabalho para desenvolvimento da mesma. Considerando a possibilidade de produção em grande escala, algumas sugestões são traçadas para futura comercialização do mesmo, como:

- Utilização do CI ADS1293, que tem baixo consumo de energia, de três canais de medições de biopotencias de alta resolução, e fornecendo novas derivações para um exame mais completo.

- Substituir o monitor OLED por um monitor sensível ao toque, de baixo consumo, para eliminação dos botões e fornecendo uma melhor interação entre homem e máquina.

- Utilização de um *slot* de cartão de memória micro (µSD), para ocupar menos espaço físico, desejável em um dispositivo portátil.

- Integrar todos os dispositivos em uma única placa de circuito impresso (PCB), eliminando circuitos desnecessários para a aplicação e optar por dispositivos de baixo consumo de energia.

- Aprimorar o desenvolvimento do firmware, evitando ociosidade do processador, para eficiência energética.

REFERÊNCIAS

ALIEXPRESS. **AliExpress**, 8 Outubro 2018. Disponivel em: <https://pt.aliexpress.com/item/Single-Lead-AD8232-Heart-Rate-Monitor-ECG-Developemt-Kit-Arduino-Compatible/32658847557.html>.

ANALOG DEVICES. Eletronic Components Datasheet Search. **Alldatasheets.com**, 8 Outubro 2018. Disponivel em: <http://www.alldatasheet.com/datasheet-pdf/pdf/527942/AD/AD8232.html>.

AULA DE ANATOMIA. Coração. **Aula de Anatomia**, 24 Abril 2017. Disponivel em: <http://www.auladeanatomia.com/novosite/sistemas/sistema-cardiovascular/coracao/>. Acesso em: 24 abril 2017.

BAÚ DA ELETRÔNICA. Módulo Cartão SD Card. **Baú da Eletrônica**, 8 Outubro 2018. Disponivel em: <http://www.baudaeletronica.com.br/modulo-sd-card.html?gclid=Cj0KCQjwgOzdBRDlARIsAJ6_HNk0p2cee-CIyarfo_HrtcTDlaIAMiQgZsocrVETpVNGY_a__tizecoaAlfhEALw_wcB>.

BERTONHA, E. G. Gravador Holter Digital. **Dissertação de Mestrado**, 1993.

DEPARTAMENTO DE FISIOLOGIA E FARMACOLOGIA. Universidade Federal Fluminense. **O eletrocardiograma**, 2006. Disponivel em: <http://www.uff.br/fisio6/aulas/aula_10/topico_09.htm>. Acesso em: 26 abril 2017.

DUARTE, D. O ECG-Curso de Eletrocardiografia. **O ECG**, 2017. Disponivel em: <http://ecg.med.br/derivacoes.asp>. Acesso em: 19 abril 2017.

DYNAPACK. DocPlayer.net. **DocPlayer**. Disponivel em: <https://docplayer.net/36583360-Material-safety-data-sheet-dell.html>. Acesso em: 15 out. 2018.

ECGPEDIA. ECGpedia.org. **ECGpedia**, 2014. Disponivel em: <http://en.ecgpedia.org/index.php?title=Basics>. Acesso em: 12 maio 2017.

ENGENHARIA Biomédica. **PEB**, 17 Abril 2017. Disponivel em: <http://www.peb.ufrj.br/eb.htm>. Acesso em: 18 abril 2017.

FYE, W. B. A history of the origin, evolution and impact of electrocardiography. **Am J Cardiol.**, p. 73:937-49., 1994.

GIFFONI, R. T.; TORRES, R. M. Breve história da eletrocardiografia. **Revista Médica de Minas Gerais**, p. 263-270, 2010.

GRUPI, C. J.; BRITO, F. ; UCHIDA, A. H. Eletrocardiograma de Longa Duração: o Sistema Holter - Parte I. **Revista Latino-Americana de Marcapasso e Arritmia**, p. 86-92, 1999.

GUYTON, A. C.; HALL, J. E. **Tratado de Fisiologia Médica**. Rio de Janeiro: Elsevier, 2006.

MIT-BIH Arrhythmia Database. **Physionet.org**, 16 Outubro 2018. Disponivel em: <https://physionet.org/physiobank/database/mitdb/>.

NATIONAL INSTRUMENTS CORPORATION. National Instruments. **NI**. Disponivel em: <http://www.ni.com/newsletter/51141/pt/>. Acesso em: 25 Outubro 2018.

RESENDE, L. O. et al. ANÁLISE DO ELETROCARDIOGRAMA (ECG) NORMAL – ASPECTOS ELÉTRICOS E FISIOLÓGICOS EM UMA ABORDAGEM INTERDISCIPLINAR, 2008.

SILVA, D. Bomba de sódio e potássio. **Estudo Prático**, 16 Abril 2017. Disponivel em: <http://www.estudopratico.com.br/bomba-de-sodio-e-potassio/>. Acesso em: 17 abril 2017.

STM32. **wiki.stm32duino.com**, 2 Outubro 2018. Disponivel em: <https://wiki.stm32duino.com/index.php?title=Blue_Pill>.

STMICROELETRONICS. STMicroeletronics. **ST**. Disponivel em: <https://www.st.com/content/ccc/resource/technical/document/reference_manual/59/

b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf>. Acesso em: 12 Outubro 2018.

TOMPKINS, W. J. **Biomedical digital signal processing**. [S.l.]: Prentice Hall, 1993.

TORTORA, G. J.; DERRICKSON, B. **Corpo Humano:** Fundamentos de Anatomia e Fisiologia. Porto Alegre: artmed, 2017.

TURCHIELLO, G. D. M. PLATAFORMA VISUAL DE PROCESSAMENTO DIGITAL DE SINAIS BIOMÉDICOS. **Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2014**, 2014. Disponivel em: <https://repositorio.ufsc.br/bitstream/handle/123456789/134755/334174.pdf?sequence=1&isAllowed=y>. Acesso em: 10 maio 2017.

WHO. Cardiovascular diseases. **Word Health Organization**, 2017. Disponivel em: <http://who.int/cardiovascular_diseases/en/>. Acesso em: 29 abril 2017.

YOUW ANG ELECTRONICS CO. datasheetspdf.com. Disponivel em: <https://datasheetspdf.com/pdf-file/695285/YOUWANG/UTC7805/1>. Acesso em: 26 Outubro 2018.

ANEXO A – Algoritmo desenvolvido para a aplicação

```
#include <RTClock.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX_AS.h>
#include <Adafruit_SSD1306_STM32.h>
#include <SdFat.h>

#define OLED_RESET 21
Adafruit_SSD1306 display(OLED_RESET);

#define LED PC13

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

typedef struct TimeElements
{
  uint8_t Second;
  uint8_t Minute;
  uint8_t Hour;
  uint8_t Wday;   // Day of week, sunday is day 1
  uint8_t Day;
  uint8_t Month;
  uint8_t Year;   // Offset from 1970;
} TimeElements ;

//************Button*****************//
int P1 = PA0; // Button -
int P2 = PA1; // Button +
int P3 = PA2; // Button Entra
int P4 = PA3; // Button Volta

//************Variables**************//
int hourupg = 1;
int minupg = 1;
int yearupg;
int monthupg = 1;
int dayupg = 1;
int menu = 0;
int flag = 0;
int flagrtcok = 0;
int numGrav = 0;

const int chipSelect = 16;

char str0[16];
char str1[16];
char str2[16];
char st0[16];
char st1[16];

String grava[32];


uint32_t      dateTime_t;
```

```
TimeElements  dateTime;

RTClock rt (RTCSEL_LSE);  // Initialise RTC with LSE

const uint8_t ADC_DIM = 2;//4
struct data_t {
  unsigned long time;
  unsigned short adc[ADC_DIM];
};
// User data functions.  Modify these functions for your data items.
int channel[ADC_DIM] = {4, 17};

// Acquire a data record.
void acquireData(data_t* data) {
  data->time = micros();
  for (int i = 0; i < ADC_DIM; i++) {
    data->adc[i] = analogRead(channel[i]);
  }
}

// Print a data record.
void printData(Print* pr, data_t* data) {
  pr->print(data->time);
  for (int i = 0; i < ADC_DIM; i++) {
    pr->write(',');
    pr->print(data->adc[i]);
  }
  pr->println();
}

// Print data header.
void printHeader(Print* pr) {
  String dataString0 = "";

  dataString0 += String(st0);
  dataString0 += String(st1);

  pr->println(dataString0);
  pr->print(F("time"));
  for (int i = 0; i < ADC_DIM; i++) {
    pr->print(F(",adc"));
    pr->print(i);
  }
  pr->println();
}
//==============================================================================
==
// Start of configuration constants.
//==============================================================================
==
//Interval between data records in microseconds.
const uint32_t LOG_INTERVAL_USEC = 4000;
//------------------------------------------------------------------------------
// File definitions.
//
// Maximum file size in blocks.
// The program creates a contiguous file with FILE_BLOCK_COUNT 512 byte blocks.
// This file is flash erased using special SD commands.  The file will be
// truncated if logging is stopped early.
```

```
const uint32_t FILE_BLOCK_COUNT = 256000;

// log file base name.  Must be six characters or less.
#define FILE_BASE_NAME "data"
//------------------------------------------------------------------------------
// Buffer definitions.
//
// The logger will use SdFat's buffer plus BUFFER_BLOCK_COUNT additional
// buffers.
//
#ifndef RAMEND
// Assume ARM. Use total of nine 512 byte buffers.
const uint8_t BUFFER_BLOCK_COUNT = 8;
//
#elif RAMEND < 0X8FF
#error Too little SRAM
//
#elif RAMEND < 0X10FF
// Use total of two 512 byte buffers.
const uint8_t BUFFER_BLOCK_COUNT = 1;
//
#elif RAMEND < 0X20FF
// Use total of five 512 byte buffers.
const uint8_t BUFFER_BLOCK_COUNT = 4;
//
#else  // RAMEND
// Use total of 13 512 byte buffers.
const uint8_t BUFFER_BLOCK_COUNT = 12;
#endif  // RAMEND
//==============================================================================
==
// End of configuration constants.
//==============================================================================
==
// Temporary log file.  Will be deleted if a reset or power failure occurs.
#define TMP_FILE_NAME "tmp_log.bin"

// Size of file base name.  Must not be larger than six.
const uint8_t BASE_NAME_SIZE = sizeof(FILE_BASE_NAME) - 1;

SdFat sd;

SdBaseFile binFile;

char binName[13] = FILE_BASE_NAME "00.bin";

// Number of data records in a block.
const uint16_t DATA_DIM = (512 - 4) / sizeof(data_t);

//Compute fill so block size is 512 bytes.  FILL_DIM may be zero.
const uint16_t FILL_DIM = 512 - 4 - DATA_DIM * sizeof(data_t);

struct block_t {
  uint16_t count;
  uint16_t overrun;
  data_t data[DATA_DIM];
  uint8_t fill[FILL_DIM];
};
```

```
const uint8_t QUEUE_DIM = BUFFER_BLOCK_COUNT + 2;

block_t* emptyQueue[QUEUE_DIM];
uint8_t emptyHead;
uint8_t emptyTail;

block_t* fullQueue[QUEUE_DIM];
uint8_t fullHead;
uint8_t fullTail;

// Advance queue index.
inline uint8_t queueNext(uint8_t ht) {
  return ht < (QUEUE_DIM - 1) ? ht + 1 : 0;
}
//==============================================================================
==
// Error messages stored in flash.
#define error(msg) errorFlash(F(msg))
//------------------------------------------------------------------------------
void errorFlash(const __FlashStringHelper* msg) {
  sd.errorPrint(msg);
  //fatalBlink();
}

// Convert binary file to csv file.
void binaryToCsv() {
  uint8_t lastPct = 0;
  block_t block;
  uint32_t t0 = millis();
  uint32_t syncCluster = 0;
  SdFile csvFile;
  char csvName[13];
  float value1;

  if (!binFile.isOpen()) {
    display.println("No current binary file");
    display.display();
    return;
  }
  binFile.rewind();
  // Create a new csvFile.
  strcpy(csvName, binName);
  strcpy(&csvName[BASE_NAME_SIZE + 3], "csv");

  if (!csvFile.open(csvName, O_WRITE | O_CREAT | O_TRUNC)) {
    error("open csvFile failed");
  }
  sprintf(str0, "Writing: %s", csvName);

  display.println(str0);
  display.display();
  display.println(" - type any character to stop");
  printHeader(&csvFile);
  uint32_t tPct = millis();
  while (flagrtcok == 1 && binFile.read(&block, 512) == 512) {
    uint16_t i;
    if (block.count == 0) {
      break;
    }
```

```
    if (block.overrun) {
      csvFile.print(F("OVERRUN,"));
      csvFile.println(block.overrun);
    }
    for (i = 0; i < block.count; i++) {
      printData(&csvFile, &block.data[i]);
    }
    if (csvFile.curCluster() != syncCluster) {
      csvFile.sync();
      syncCluster = csvFile.curCluster();
    }
    if ((millis() - tPct) > 1000) {
      uint8_t pct = binFile.curPosition() / (binFile.fileSize() / 100);
      if (pct != lastPct) {
        tPct = millis();
        lastPct = pct;
        sprintf(str1, "  %u %", pct);
        display.clearDisplay();
        display.setCursor(0, 0);
        display.println(str1);
        display.display();
      }
    }
    if (Serial.available()) {
      break;
    }
  }
  csvFile.close();
  value1 = (0.001 * (millis() - t0));
  sprintf(str2, "Done: %f seconds" , value1);
  display.print(str2);
  display.display();
  PrintMomento();
}

// log data
// max number of blocks to erase per erase call
uint32_t const ERASE_SIZE = 262144L;
void logData() {
  uint32_t bgnBlock, endBlock;

  // Allocate extra buffer space.
  block_t block[BUFFER_BLOCK_COUNT];
  block_t* curBlock = 0;
  display.println();
  display.display();

  // Find unused file name.
  if (BASE_NAME_SIZE > 6) {
    error("FILE_BASE_NAME too long");
  }
  while (sd.exists(binName)) {
    if (binName[BASE_NAME_SIZE + 1] != '9') {
      binName[BASE_NAME_SIZE + 1]++;
    } else {
      binName[BASE_NAME_SIZE + 1] = '0';
      if (binName[BASE_NAME_SIZE] == '9') {
        error("Can't create file name");
      }
```

```
    binName[BASE_NAME_SIZE]++;
  }
}
// Delete old tmp file.
if (sd.exists(TMP_FILE_NAME)) {
  display.clearDisplay();
  display.println(F("Deleting tmp file"));
  display.display();
  if (!sd.remove(TMP_FILE_NAME)) {
    error("Can't remove tmp file");
  }
}
// Create new file.
display.setCursor(0, 0);
display.clearDisplay();
display.println(F("Creating new file"));
display.display();

binFile.close();
if (!binFile.createContiguous(sd.vwd(),
                        TMP_FILE_NAME, 512 * FILE_BLOCK_COUNT)) {
  error("createContiguous failed");
}
// Get the address of the file on the SD.
if (!binFile.contiguousRange(&bgnBlock, &endBlock)) {
  error("contiguousRange failed");
}
// Use SdFat's internal buffer.
uint8_t* cache = (uint8_t*)sd.vol()->cacheClear();
if (cache == 0) {
  error("cacheClear failed");
}

// Flash erase all data in the file.
display.println(F("Erasing all data"));
display.display();

uint32_t bgnErase = bgnBlock;
uint32_t endErase;
while (bgnErase < endBlock) {
  endErase = bgnErase + ERASE_SIZE;
  if (endErase > endBlock) {
    endErase = endBlock;
  }
  if (!sd.card()->erase(bgnErase, endErase)) {
    error("erase failed");
  }
  bgnErase = endErase + 1;
}
// Start a multiple block write.
if (!sd.card()->writeStart(bgnBlock, FILE_BLOCK_COUNT)) {
  error("writeBegin failed");
}
// Initialize queues.
emptyHead = emptyTail = 0;
fullHead = fullTail = 0;

// Use SdFat buffer for one block.
emptyQueue[emptyHead] = (block_t*)cache;
```

```
emptyHead = queueNext(emptyHead);

// Put rest of buffers in the empty queue.
for (uint8_t i = 0; i < BUFFER_BLOCK_COUNT; i++) {
  emptyQueue[emptyHead] = &block[i];
  emptyHead = queueNext(emptyHead);
}
display.println("Logging - type any character to stop");
display.display();

// Wait for Serial Idle.
//Serial.flush();
delay(10);
uint32_t bn = 0;
uint32_t t0 = millis();
uint32_t t1 = t0;
uint32_t overrun = 0;
uint32_t overrunTotal = 0;
uint32_t count = 0;
uint32_t maxLatency = 0;
int32_t diff;
// Start at a multiple of interval.
uint32_t logTime = micros() / LOG_INTERVAL_USEC + 1;
logTime *= LOG_INTERVAL_USEC;
bool closeFile = false;
while (1) {
  // Time for next data record.
  logTime += LOG_INTERVAL_USEC;
  if (digitalRead(P4)) {
    closeFile = true;
  }

  if (closeFile) {
    if (curBlock != 0 && curBlock->count >= 0) {
      // Put buffer in full queue.
      fullQueue[fullHead] = curBlock;
      fullHead = queueNext(fullHead);
      curBlock = 0;
    }
  } else {
    if (curBlock == 0 && emptyTail != emptyHead) {
      curBlock = emptyQueue[emptyTail];
      emptyTail = queueNext(emptyTail);
      curBlock->count = 0;
      curBlock->overrun = overrun;
      overrun = 0;
    }
    do {
      diff = logTime - micros();
    } while (diff > 0);
    if (diff < -10) {
      error("LOG_INTERVAL_USEC too small");
    }
    if (curBlock == 0) {
      overrun++;
    } else {
      acquireData(&curBlock->data[curBlock->count++]);
      if (curBlock->count == DATA_DIM) {
        fullQueue[fullHead] = curBlock;
```

```
        fullHead = queueNext(fullHead);
        curBlock = 0;

        if (digitalRead(P3)){
        GravaMomento();
        }
      }
    }
  }

  if (fullHead == fullTail) {
    // Exit loop if done.
    if (closeFile) {
      break;
    }
  } else if (!sd.card()->isBusy()) {
    // Get address of block to write.
    block_t* pBlock = fullQueue[fullTail];
    fullTail = queueNext(fullTail);
    // Write block to SD.
    uint32_t usec = micros();
    if (!sd.card()->writeData((uint8_t*)pBlock)) {
      error("write data failed");
    }
    usec = micros() - usec;
    t1 = millis();
    if (usec > maxLatency) {
      maxLatency = usec;
    }
    count += pBlock->count;

    // Add overruns and possibly light LED.
    if (pBlock->overrun) {
      overrunTotal += pBlock->overrun;
  /*    if (ERROR_LED_PIN >= 0) {
        digitalWrite(ERROR_LED_PIN, HIGH);
      }*/
    }
    // Move block to empty queue.
    emptyQueue[emptyHead] = pBlock;
    emptyHead = queueNext(emptyHead);
    bn++;
    if (bn == FILE_BLOCK_COUNT) {
      // File full so stop
      break;
    }
  }
}
if (!sd.card()->writeStop()) {
  error("writeStop failed");
}
// Truncate file if recording stopped early.
if (bn != FILE_BLOCK_COUNT) {
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Truncating file"));
  display.display();
  flag = 1;
  menu = 1;
```

```
    if (!binFile.truncate(512L * bn)) {
      error("Can't truncate file");
    }
  }
  if (!binFile.rename(sd.vwd(), binName)) {
    error("Can't rename file");
  }
  display.println(F("Done"));
}
//------------------------------------------------------------------------------

void setup()
{
  pinMode(LED, OUTPUT);

  pinMode(P1, INPUT);
  pinMode(P2, INPUT);
  pinMode(P3, INPUT);
  pinMode(P4, INPUT);

  // Some delay waiting serial ready
  digitalWrite(LED, HIGH);
  delay(3000);
  digitalWrite(LED, LOW);

  Serial.begin(115200);

  //SetupModules();
  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3D (for the 128x64)
  // init done
  // Show image buffer on the display hardware.
  // Since the buffer is intialized with an Adafruit splashscreen
  // internally, this will display the splashscreen.
  display.display();
  delay(1000);

  // Clear the buffer.
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);

  display.println("Inicializando cartao de memoria");
  display.display();
  delay(1000);
  display.clearDisplay();

  //* see if the card is present and can be initialized:
  sprintf(str0, "Records/block: %d", DATA_DIM);
  display.println(str0);
  display.display();

  if (sizeof(block_t) != 512) {
    error("Invalid block size");
  }
  // initialize file system.
  if (!sd.begin(chipSelect, SPI_FULL_SPEED)) {
    sd.initErrorPrint();
```

```
    display.println("card not inicialized");
    display.display();
    //não faça nada
    while(1);
  }

  display.println("card initialized.");
  display.display();
  delay(1000);
  display.clearDisplay();

  menu = 0;
  flag = 0;

  display.setTextSize(2);
}

void loop()
{
  if (flagrtcok == 0)SetRtc();

  if (flagrtcok == 1)Startlog();
}

void Startlog() {
  Buttons0();
  if (menu == 0)
  {
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 0);
    breakTime(rt.getTime(), dateTime);
    sprintf(str0, "Hora %u:%u:%u  ", dateTime.Hour, dateTime.Minute, dateTime.Second);
    display.println(str0);
    display.display();
    sprintf(str1, "Data %u/%u/%u", dateTime.Day, dateTime.Month, dateTime.Year + 1970);
    display.println(str1);
    display.display();
    display.println("Para gravar, ajuste a hora. Prosseguir?");
    display.display();
    display.println("+ sim / - nao");
    display.display();

    while (menu == 0 && flag == 2) {

      breakTime(rt.getTime(), dateTime);
      sprintf(st0,"Hora %u:%u:%u  ", dateTime.Hour, dateTime.Minute, dateTime.Second);
      sprintf(st1,"Data %u/%u/%u", dateTime.Day, dateTime.Month, dateTime.Year+1970);
      logData();

    }
  }

  if (menu == 1) {
    binaryToCsv();
    menu = 0;
  }
  if (flag == 0) {
    display.clearDisplay();
```

```
    flagrtcok = 0;
    menu = 0;
    flag = 0;
    display.setTextSize(2);
  }
}

void SetRtc() {
  Buttons0();
  // in which subroutine should we go?
  if (flag == 0) {
    HelpScreen();
  }
  if (menu == 0)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Mostra");
    display.display();
    while (menu == 0 && flag == 2) {
      RTCdisplay(); // void DisplayDateTime
    }
  }
  if (menu == 1)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Seta Hora");
    display.display();
    while (menu == 1 && flag == 2) {
      DisplaySetHour();
    }
  }
  if (menu == 2)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Seta Min");
    display.display();
    while (menu == 2 && flag == 2) {
      DisplaySetMinute();
    }
  }
  if (menu == 3)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Seta Ano");
    display.display();
    while (menu == 3 && flag == 2) {
      DisplaySetYear();
    }
  }
  if (menu == 4)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Seta Mes");
    display.display();
```

```
    while (menu == 4 && flag == 2) {
      DisplaySetMonth();
    }
  }
  if (menu == 5)
  {
    display.clearDisplay();
    display.setCursor(5, 0);
    display.println("Seta Dia");
    display.display();
    while (menu == 5 && flag == 2) {
      DisplaySetDay();
    }
  }
  if (menu == 6)
  {
    StoreAgg();
    menu = 0;
    flagrtcok = 1;
    flag = 1;
    delay(500);
  }
}

void Buttons0() {
  // check if you press the SET button and increase the menu index
  if (digitalRead(P2) == HIGH)
  {
    menu = menu + 1;
    if (menu == 7) {
      menu = 0;
    }
  }
  if (digitalRead(P1) == HIGH)
  {
    menu = menu - 1;
    if (menu == -1) {
      menu = 5;
    }
  }
  if (digitalRead(P3) == HIGH)
  {
    flag = flag + 1;
    if (flag == 3) {
      flag = 1;
    }
  }
  if (digitalRead(P4) == HIGH)
  {
    flag = flag - 1;
    if (flag == -1) {
      flag = 0;
    }
  }
}

void GravaMomento(){
  String dataString0 = "";
  char str0[16];
```

```
  char str1[16];

  display.clearDisplay();
  display.setTextSize(1);

  breakTime(rt.getTime(), dateTime);
  sprintf(str0,"Hora %u:%u:%u  ", dateTime.Hour, dateTime.Minute, dateTime.Second);
  dataString0 += String(str0);

  sprintf(str1,"Data %u/%u/%u", dateTime.Day, dateTime.Month, dateTime.Year+1970);
  dataString0 += String(str1);

  grava[numGrav] = dataString0;
  numGrav++;

}

void PrintMomento(){

  SdFile txtFile;
  char txtName[13];
  int i = 0;
  // Create a new txtFile.
  strcpy(txtName, binName);
  strcpy(&txtName[BASE_NAME_SIZE + 3], "txt");

  if (!txtFile.open(txtName, O_WRITE | O_CREAT | O_TRUNC)) {
    error("open csvFile failed");
  }

  for(i=0; i <= numGrav; i++){
     txtFile.println(grava[numGrav]);
     }
  txtFile.close();
  numGrav=0;
}

void HelpScreen() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 20);
  display.println("Menos Mais Entra Sai");
  display.display();
  display.setTextSize(2);
  display.println("-  +  >  <");
  display.display();
  delay(2000);
  flag = 1;
}

void RTCdisplay()
{
  // Display RCT time
  display.clearDisplay();
  display.setCursor(10, 0);
  display.println("Hora/Data");
  display.display();
  breakTime(rt.getTime(), dateTime);
  sprintf(str0, "%u:%u:%u", dateTime.Hour, dateTime.Minute, dateTime.Second);
```

```
  sprintf(str1, "%u/%u/%u", dateTime.Day, dateTime.Month, dateTime.Year + 1970);
  display.println(str0);
  display.display();
  display.println(str1);
  display.display();
  digitalWrite(LED, !digitalRead(LED));
  if (digitalRead(P4) == HIGH)
  {
    flag = - 1;
  }
}

void DisplaySetHour()
{
  // time setting

  if (digitalRead(P2))
  {
    if (hourupg == 23)
    {
      hourupg = 0;
    }
    else
    {
      hourupg = hourupg + 1;
    }
  }
  if (digitalRead(P1))
  {
    if (hourupg == 0)
    {
      hourupg = 23;
    }
    else
    {
      hourupg = hourupg - 1;
    }
  }
  if (digitalRead(P4))
  {
    flag = flag - 1;
  }
  //display.setCursor(0,0);
  display.clearDisplay();
  display.setCursor(20, 0);
  display.println("Hora:");
  display.display();
  display.println(hourupg);
  display.display();
  dateTime.Hour = hourupg;
}

void DisplaySetMinute()
{

  if (digitalRead(P2))
  {
    if (minupg == 59)
    {
```

```
      minupg = 0;
    }
    else
    {
      minupg = minupg + 1;
    }
  }
  if (digitalRead(P1))
  {
    if (minupg == 0)
    {
      minupg = 59;
    }
    else
    {
      minupg = minupg - 1;
    }
  }
  if (digitalRead(P4))
  {
    flag = flag - 1;
  }
  //display.setCursor(0,0);
  display.clearDisplay();
  display.setCursor(20, 0);
  display.println("Minuto:");
  display.display();
  display.println(minupg);
  display.display();
  dateTime.Minute = minupg;
}

void DisplaySetYear()
{

  if (digitalRead(P2))
  {
    yearupg = yearupg + 1;
  }
  if (digitalRead(P1))
  {
    yearupg = yearupg - 1;
  }
  if (digitalRead(P4))
  {
    flag = flag - 1;
  }
  //display.setCursor(0,0);
  display.clearDisplay();
  display.setCursor(20, 0);
  display.println("Ano:");
  display.display();
  //display.setCursor(0,1);
  display.println(yearupg + 1970);
  display.display();
  dateTime.Year = yearupg;
  delay(100);
}
```

```
void DisplaySetMonth()
{

  if (digitalRead(P2))
  {
    if (monthupg == 12)
    {
      monthupg = 1;
    }
    else
    {
      monthupg = monthupg + 1;
    }
  }
  if (digitalRead(P1))
  {
    if (monthupg == 1)
    {
      monthupg = 12;
    }
    else
    {
      monthupg = monthupg - 1;
    }
  }
  if (digitalRead(P4))
  {
    flag = flag - 1;
  }
  //display.setCursor(0,0);
  display.clearDisplay();
  display.setCursor(20, 0);
  display.println("Mes:");
  display.display();
  //display.setCursor(0,1);
  display.println(monthupg);
  display.display();
  dateTime.Month = monthupg;
}

void DisplaySetDay()
{
  if (digitalRead(P2))
  {
    if (dayupg == 31)
    {
      dayupg = 1;
    }
    else
    {
      dayupg = dayupg + 1;
    }
  }
  if (digitalRead(P1))
  {
    if (dayupg == 1)
    {
      dayupg = 31;
    }
```

```
    else
    {
      dayupg = dayupg - 1;
    }
  }
  if (digitalRead(P4))
  {
    flag = flag - 1;
  }
  //display.setCursor(0,0);
  display.clearDisplay();
  display.setCursor(20, 0);
  display.println("Dia:");
  display.display();
  //display.setCursor(0,1);
  display.println(dayupg);
  display.display();
  dateTime.Day = dayupg;
}

void StoreAgg()
{
  display.clearDisplay();
  display.setCursor(10, 0);
  display.print("SAVING IN");
  display.display();
  //display.setCursor(0,1);
  display.print("PROGRESS");
  display.display();
  dateTime_t = makeTime(dateTime);    // Store compilation time in "unix" format
  rt.setTime(dateTime_t);
}

/*=====================================================================================
*/
/* Functions to convert to and from system time (From https://github.com/PaulStoffregen/Time */

// Leap year calulator expects year argument as years offset from 1970
#define LEAP_YEAR(Y)  ( ((1970+Y)>0) && !((1970+Y)%4) && ( ((1970+Y)%100) || !((1970+Y)%400) )
)
#define SECS_PER_MIN  (60UL)
#define SECS_PER_HOUR (3600UL)
#define SECS_PER_DAY  (SECS_PER_HOUR * 24UL)

//static const uint8_t monthDays[]={31,28,31,30,31,30,31,31,30,31,30,31}; // API starts months from 1,
this array starts from 0

void breakTime(uint32_t timeInput, struct TimeElements &tm) {
  // Break the given time_t into time components
  // This is a more compact version of the C library localtime function
  // Note that year is offset from 1970 !!!

  uint8_t year;
  uint8_t month, monthLength;
  uint32_t time;
  unsigned long days;

  time = (uint32_t)timeInput;
  tm.Second = time % 60;
```

```
  time /= 60; // Now it is minutes
  tm.Minute = time % 60;
  time /= 60; // Now it is hours
  tm.Hour = time % 24;
  time /= 24; // Now it is days
  tm.Wday = ((time + 4) % 7) + 1;  // Sunday is day 1

  year = 0;
  days = 0;
  while ((unsigned)(days += (LEAP_YEAR(year) ? 366 : 365)) <= time) {
    year++;
  }
  tm.Year = year; // Year is offset from 1970

  days -= LEAP_YEAR(year) ? 366 : 365;
  time  -= days; // Now it is days in this year, starting at 0

  days = 0;
  month = 0;
  monthLength = 0;
  for (month = 0; month < 12; month++) {
    if (month == 1) { // February
      if (LEAP_YEAR(year)) {
        monthLength = 29;
      } else {
        monthLength = 28;
      }
    } else {
      monthLength = monthDays[month];
    }

    if (time >= monthLength) {
      time -= monthLength;
    } else {
      break;
    }
  }
  tm.Month = month + 1;  // Jan is month 1
  tm.Day = time + 1;     // Day of month
}

uint32_t makeTime(struct TimeElements &tm) {
  // Assemble time elements into "unix" format
  // Note year argument is offset from 1970

  int i;
  uint32_t seconds;

  // Seconds from 1970 till 1 jan 00:00:00 of the given year
  seconds = tm.Year * (SECS_PER_DAY * 365);
  for (i = 0; i < tm.Year; i++) {
    if (LEAP_YEAR(i)) {
      seconds +=  SECS_PER_DAY;   // Add extra days for leap years
    }
  }

  // Add days for this year, months start from 1
  for (i = 1; i < tm.Month; i++) {
    if ( (i == 2) && LEAP_YEAR(tm.Year)) {
```

```
      seconds += SECS_PER_DAY * 29;
    } else {
      seconds += SECS_PER_DAY * monthDays[i - 1]; // MonthDay array starts from 0
    }
  }
  seconds += (tm.Day - 1) * SECS_PER_DAY;
  seconds += tm.Hour * SECS_PER_HOUR;
  seconds += tm.Minute * SECS_PER_MIN;
  seconds += tm.Second;
  return (uint32_t)seconds;
}
```