UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Karoline Teixeira Pereira

Mapeamento de Trajetórias de Máquinas Pesadas Usando Arduíno e GPS

São Carlos

2017

Karoline Teixeira Pereira

Mapeamento de Trajetórias de Máquinas Pesadas Usando Arduíno e GPS

Monografia apresentada ao Curso de Engenharia Elétrica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Rogério Andrade Flauzino

São Carlos

2017

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes e Seção Técnica de Informática, EESC/USP com os dados fornecidos pelo(a) autor(a).

Pereira, Karoline
P436m Mapeamento de Trajetórias de Máquinas Pesadas Usando Arduino e GPS / Karoline Pereira; orientador Rogério Flauzino. São Carlos, 2017.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Arduino. 2. GPS. 3. Trajetória. 4. Mapeamento. I. Título.

Bibliotecário responsável pela estrutura de catalogação da publicação:
Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Karoline Teixeira Pereira

Título: "Mapeamento de trajetórias de máquinas pesadas usando arduíno e GPS"

Trabalho de Conclusão de Curso defendido e aprovado em 01 / 12 / 2017 ,

com NOTA 9,0 (Nove, Zero), **pela Comissão Julgadora:**

Prof. Associado Rogério Andrade Flauzino - Orientador - SEL/EESC/USP

Mestre Ricardo Luís Balieiro - Doutorando - SEL/EESC/USP

Mestre Murilo Eduardo Casteroba Bento - Doutorando - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado a minha mãe Noédia Pereira sem a qual nada disso seria possível ao meu pai Antônio Pereira por todo apoio e incentivo, ao meu irmão Antônio e minha irmã Elza Maria. Dedico também ao meu namorado Giácomo por estar sempre ao meu lado nesses anos tão difíceis

AGRADECIMENTOS

Agradeço primeiramente a minha família por todo apoio, suporte e confiança a mim depositadas, fundamentais no decorrer da minha vida e graduação.

Ao meu namorado, amigo e companheiro, Giácomo Mergulhão Estronioli, por todo apoio, força, paciência e compreensão durante todos esses anos de graduação.

Aos meus amigos Murilo Rocha e Rafael Magossi, por todo conhecimento compartilhado, todo auxílio, paciência e amizade. Sem eles essa trajetória seria ainda mais difícil.

Ao Prof. Dr. Rogério Andrade Flauzino pela orientação, suporte e oportunidade no decorrer do desenvolvimento deste trabalho.

Aos inúmeros colegas de classe pela convivência e pelo conhecimento compartilhado durante esses anos de graduação.

À todos os professores e funcionários do campus, em especial aos da Escola de Engenharia de São Carlos que nos fornecem diariamente oportunidade e qualidade nos aprendizados oferecidos.

À autogestão do alojamento, aos funcionários e funcionárias da assistência social que oferecem condições para que todos possam permanecer e dar continuidade ao ensino de qualidade oferecido pela Universidade de São Paulo.

"O homem não teria alcançado o possível se,
repetidas vezes não tivesse tentado o impossível"
Max Weber.

RESUMO

Pereira, K. T. **Mapeamento de Trajetórias de Máquinas Pesadas Usando Arduíno e GPS**. 2017. 59p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

Neste trabalho é apresentado o desenvolvimento e implementação de um dispositivo para o mapeamento da trajetória de máquinas de grande porte ainda dentro do fluxo de produção utilizando Arduíno Mega e Shield GPS. De forma que com o auxilio da ferramente Lean manufacturing, esses dados possibilitem a realização de um estudo referente ao excesso de movimentação de máquinas e de alternativas para otimização do fluxo de testes, com a finalidade de reduzir o deslocamento dos veículos e o tempo gasto no processo.

Palavras-chave: Arduino. GPS. Trajetória. Mapeamento. Lean manufacturing.

ABSTRACT

Pereira, K. T. **Mapping of Heavy Machinery Trajectories in Test Using Arduíno and GPS**. 2017. 59p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

This paper presents the development and implementation of a device to map the trajectory of large machines still within the production flow using Arduino Mega and Shield GPS. Thus, with the help of the Lean manufacturing tool, these data make it possible to carry out a study regarding the excessive movement of machines and alternatives to optimize the flow of tests, in order to reduce vehicle displacement and the time spent in the process.

Keywords: Arduino. GPS.Trajectory. Mapping. Lean manufacturing.

SUMÁRIO

1 INTRODUÇÃO

A constante busca por produtos competitivos acarreta a necessidade crescente de redução de custos dentro das organizações. Com isso, a escassez da disponibilidade de recursos para investimento, faz com que o emprego de técnicas simples e de baixo custo que atuem nesse sentido sejam imprescindíveis (TECNOLOGIA, 2012). Além disso, o crescimento da abertura dos mercados torna cada vez mais presente a competitividade dos países emergentes fazendo com que seja necessário que as empresas tornem seus processos mais eficientes para assegurarem sua sobrevivência e crescimento. (LIKER; MEIER, 2006).

Nesse contexto de alta competitividade emergiu o conceito de *Lean Manufacturing* nascido na Toyota, no Japão após a segunda guerra mundial e cunhado originalmente em 1990 no livro "A Máquina que Mudou o Mundo", do autor James Wormack. Essa filosofia surgiu como fruto de um vasto estudo realizado pelo Massachusetts Institute of Technology (MIT) sobre a indústria automobilística mundial. Esse estudo mostrou as enormes diferenças em produtividade, desenvolvimento de produtos, qualidade e as vastas vantagens do Sistema Toyota de Produção. Ficando evidente que a Toyota havia desenvolvido um melhor modelo de gestão industrial, mudando consideravelmente o modo de se projetar, produzir e o relacionamento com fornecedores e clientes que eram praticados até então (LIKER; MEIER, 2006).

O *Lean Manufacturing* trata o conceito de fazer mais com menos recursos, seja de tempo, espaço, material, mão de obra, maquinário etc e ainda assim fornecer ao cliente aquilo que ele deseja. A filosofia lean é uma ferramenta para auxiliar na detecção de atividades que não agregam valor para o cliente, para que assim possam ser estudadas maneiras de eliminá-las, ou seja, é uma filosofia que auxilia na detecção e redução dos desperdícios.

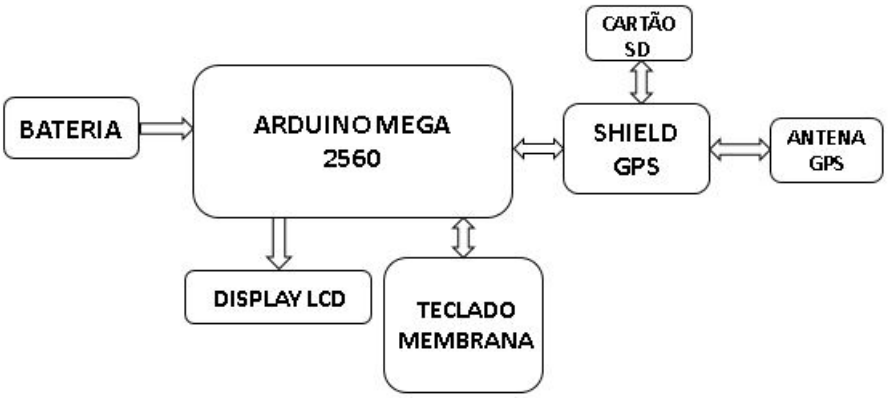
Com essa filosofia em mente, o presente projeto trata de um dispositivo eletrônico desenvolvido e implementado para realizar o mapeamento da trajetória de máquinas pesadas desde o ponto que saem da linha principal de produção e iniciam o fluxo de testes até o ponto que as máquinas estão prontas para envio ao cliente. O objetivo é que com o mapeamento da trajetória, da distância percorrida e do tempo gasto nesse percurso seja possível identificar possíveis desperdícios de movimentação e espera no processo.

Para realizar esse projeto foi usada como base uma indústria de grande porte de máquinas pesadas voltadas, principalmente, à construção civil e mineração. O dispositivo desenvolvido é composto basicamente por uma placa Arduíno Mega 2560, uma placa GPS mais antena, um carregador portátil, um teclado de membrana e um display LCD 4x20. O intuito é que, por meio do teclado o operador possa inserir o número de série da máquina

cujos dados serão coletados e por meio de um display LCD seja possível verificar os passos a serem seguidos para dar inicio a coleta. O Arduíno é o microcontrolador que ditará como os dados serão armazenados e quais serão esses dados. A placa do GPS será responsável pela conexão e coleta dos dados via satélites. Dessa forma esses dados serão gravados em um cartão de memória inserido na placa do GPS e poderão ser consultados após finalização da coleta.

Os dados recolhidos são armazenados em um arquivo .CSV cujo nome corresponde ao número de série da máquina observada. Dessa forma o cartão SD deve ser inserido em um computador onde será realizado o estudo das coordenadas GPS obtidas.

Figura 1: Esquemático do dispositivo



Fonte: Elaborado pela autora

O desenvolvimento deste trabalho inicia-se com a apresentação do modelo do Arduíno utilizado e suas vantagens frente aos outros modelos disponíveis no mercado, baseado no microcontrolador ATMega2560.

Em seguida são apresentados os periféricos utilizados, a função e objetivo de cada componente, a forma de alimentação do circuito e sua implementação em hardware.

Após a apresentação do hardware, será apresentada a lógica de programação do Software e finalizaremos esse trabalho com os testes realizados e os resultados obtidos, bem como a expectativa para trabalhos futuros.

2 HARDWARE

Para realizar o levantamento da trajetória das máquinas foi necessária a construção de um circuito de baixo custo e complexidade, capaz de ser inserido em máquinas pesadas suportando fortes trepidações, podendo ainda ser utilizado em locais parcialmente fechados, como uma fábrica e pudesse ser utilizado por qualquer pessoa. Dentro desses requisitos foram utilizados os dispositivos conforme Tabela 1

Tabela 1: Componentes de Hardware

Componentes	**Quantidade**	**Preço unitário em R$**
Placa Arduino ATMega2560	1	53,40
Modulo Arduino Shield GPS com Antena	1	149,50
Carregador Portátil Xiaomi 20000 mah	1	200,00
Display LCD 20x4	1	45,00
Quadro de distribuição Schnedier	1	50,00
Cartão de memória 16 GB	1	40,00
Potenciometro de 10K ohms	1	2,00
Resistor 10K ohms	4	0,20
Diodo 4051	4	0,20
Placa padrão 10x10 cm	1	20,00
Botão Interruptor	1	2,00
Knob Botão Potenciômetro	1	2,00
Fios Jumpers Macho Macho	40	0,50
	total	585,50

Fonte: Elaborado pela autora

2.1 Placa Arduino

Criado em 2005 na Itália, inicialmente para fins educacionais o Arduino é uma placa que possui prototipagem eletrônica de código aberto, com hardware e software livres e tem por objetivo oferecer uma ferramenta de baixo custo e simples para a criação de desenvolvimento de projetos iterativos de diversas naturezas. Ou seja, é uma plataforma utilizada para a criação de outros equipamentos (MONK, 2016).

É composto basicamente por um controlador Atmel AVR de 8 bits, com suporte entrada/saída embutido, pinos analógicos, digitais e uma interface serial ou USB.

Com código aberto pode ser modificado pelo usuário conforme a necessidade. Sua programação pode ser realizada em linguagem C/C++ utilizando o Software do Arduino (IDE) cujo ambiente é escrito em Java. A IDE pode ser executado em Windows, Mac OS X e Linux. A placa Arduino em si não possui recurso de rede, no entanto usualmente

combinam-se um ou mais Arduinos com extensões próprias chamadas shields (MONK, 2016).

Atualmente há no mercado 22 modelos distintos disponíveis de placas Arduinos conforme fig. 2, sendo os mais comuns o Arduino UNO, Mega2560, Leonardo, Pro Mini, Due e o Nano. Os dois primeiros representam os mais populares entre todos e possuem preços que diferem apenas de cerca de R$ 10,00 de um modelo para o outro. Na tabela apresentada na fig:2 podemos comparar as principais diferenças entres os modelos existentes no mercado (GENUINO, 2017).

Figura 2: Modelos de Arduinos disponíveis no mercado

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V/ 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
LilyPad/ SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	4/0	9/4	1	2	32	-	-
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8MHz	4/0	9/5	1	2.5	32	Micro	-
Mega 2560	ATmega2560	5 V / 7-12 V	16MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V/ 5V	48MHz	07/01	08/04	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	1	2	32	Regular	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/06	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	06/01	14/10	-	32	256	2 Micro	2
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/02	54/12	-	96	512	2 Micro	4
Esplora	ATmega32U4	5 V / 7-12 V	16 MHz	-	-	1	2.5	32	Micro	-
Ethernet	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/04	1	2	32	Regular	-
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/07	1	2.5	32	Micro	1
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Mini	ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/06	1	2	32	-	-
Nano	ATmega168 ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/06	0,512 1	1 2	16 32	Mini	1
Yùn	ATmega32U4 AR9331 Linux	5 V	16 MHz 400MHz	12/0	20/07	1	2.5 16MB	32 64MB	Micro	1
Arduino Robot	ATmega32u4	5 V	16 MHz	6/0	20/6	1 KB (ATmega32u4)/ 512 Kbit(I2C)	2,5KB (ATmega32u4)	32 KB (ATmega32u4) of which 4 KB used by bootloader	1	1
MKRZero	SAMD21 Cortex-M0+ 32bit low power ARM MCU	3.3 V	48 MHz	7 (ADC 08/10/12bit)/1 (DAC 10 bit)	22/12	No	32 KB	256 KB	1	1

Fonte: Genuino (2017)

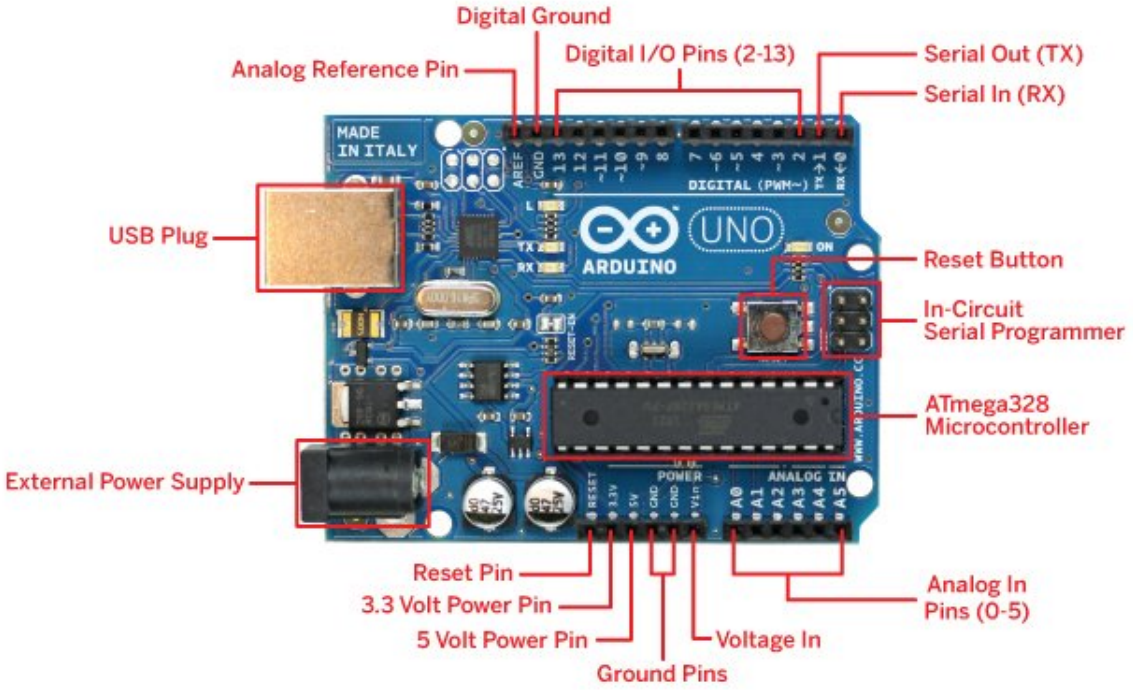
Nesse trabalho iremos detalhar os dois modelos de placas mais conhecidas e apresentar as vantagens do modelo MEGA2560 frente ao UNO.

2.1.1 Arduino UNO x Mega2560

O UNO possui microcontrolador ATmega328 com encapsulamento de 28 pinos, 20 digitais de entrada/saída, dos quais 7 podem ser também usados como saídas PWM

e 12 como entradas analógicas. Possui ainda uma conexão micro USB, clock de 16MHz, um jack de alimentação, conector ICSP e botão reset, como podemos conferir na fig:3 (PRODUCTS, 2017b).

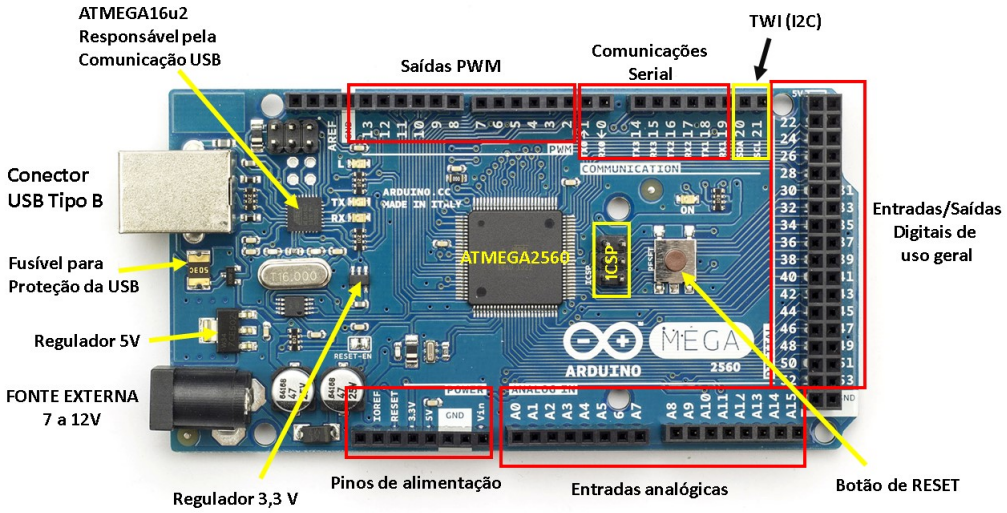
Figura 3: Placa Arduino UNO



Fonte: Dreamcatchercarriage (2017)

Já o Mega2560 é composto pelo microcontrolador ATMEGA2560, possui maior número de entradas e saídas permitindo a utilização de mais periféricos, o que possibilita a realização de projetos com maior complexidade. Tem no total 54 pinos digitais de entrada/saída, dos quais 15 podem ser utilizados como saída PWM, 4 portas seriais de hardware (UARTs), 16 entradas analógicas, cristal oscilador de 16MHz, conexão jack de alimentação, conexão USB, conector ICSP e um botão reset (PRODUCTS, 2017a). Na fig.4 é possível ver com mais detalhes as características de hardware da placa. Além da diferença no número de portas, o Mega conta com maior capacidade de memória de programa quando comparado ao UNO fazendo dele a melhor opção para projetos mais elaborados.

Figura 4: Placa Arduino Mega2560



Fonte: Souza (2014)

2.1.1.1 Microcontrolador

Como já mencionado a placa Arduino UNO utiliza o microcontrolador ATmega328 e o Arduino Mega2560 o ATMEL ATmega2560 de 8 bits e arquitetura RISC avançada. Este possui mais recursos que aquele já que conta com 256 KB de memória Flash, 8 KB de RAM e 4KB de EEPROM, contra 32 KB de memória Flash, 2 KB de RAM e 1 KB de EEPROM do UNO (PRODUCTS, 2017a) (PERFORMANCE et al., 2008).

2.1.1.2 Alimentação das placas

A alimentação das placas ocorrem de maneira semelhante nos dois modelos através de uma alimentação externa por meio de um conector Jack com positivo no centro ou por uma porta USB. Na alimentação externa a fonte deve fornecer tensão entre 6V e 20V. No entanto, é importante observar que se a tensão de alimentação for inferior a 7V pode acarretar em instabilidade, por outro lado se for alimentada com tensão acima de 12V o regulador de tensão da placa pode sobreaquecer ocasionando danos a placa (SOUZA, 2014). As placas possuem um CI para regulação de tensão quando a alimentação é externa. No caso da conexão por USB, o regulador não precisa agir, pois a própria porta já possui um circuito de proteção e com isso há uma alimentação direta.

Além disso, há nas placas um circuito para comutar, automaticamente, a fonte de alimentação entre a externa e USB. Fazendo com que se a USB for conectada e houver uma tensão no conector DC, a fonte externa que será responsável por fornecer a tensão de 5V deixando a USB apenas para comunicação com o PC.

Há ainda, um regulador de 3,3V que é responsável pelo fornecimento de tensão contínua para o circuito cujo limite máximo de corrente que pode fornecer é de 50mA (SOUZA, 2014).

2.1.1.3 Interrupções externas

No UNO há possibilidade de duas interrupções externas utilizando o pino 2 e 3, esses pinos podem ser configurados para esse fim por meio da função attachInterrupt(). Já o Mega2560 possui 6 pinos que podem ser configurados para realizar a interrupção externa, pinos 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), e 21 (interrupt 2). Com isso percebemos que o Mega, também nesse sentido, permite uma maior complexidade do projeto podendo realizar diversas interrupções simultaneamente (ATMEL et al., 2015) (PERFORMANCE et al., 2008).

2.1.1.4 Comunicação USB

A comunicação USB ocorre da mesma maneira tanto na placa do UNO quanto na Mega2560. Nos dois modelos essa comunicação entre computador e placa utiliza o microcontrolador ATMEL ATEMEGA16U2 que possibilita o upload do código gerado após compilação do programa desenvolvido pelo usuário.

Para a realização de atualizações, a gravação de firmware é feita por meio de um programador ATMEL utilizando um conector ICSP, também presente no circuito. Além disso, a sinalização do envio e recebimento de dados é realizada por dois leds presentes no microcontrolador denominados TX e RX (PRODUCTS, 2017a).

Com a finalidade de facilitar a utilização do Arduino, o pino 13 do ATMEGA16U2 é conectado ao circuito de RESET do ATMEGA2560. Dessa forma, quando pressionado o botão upload da IDE a entrada em modo bootloader ocorre automaticamente (SOUZA, 2014).

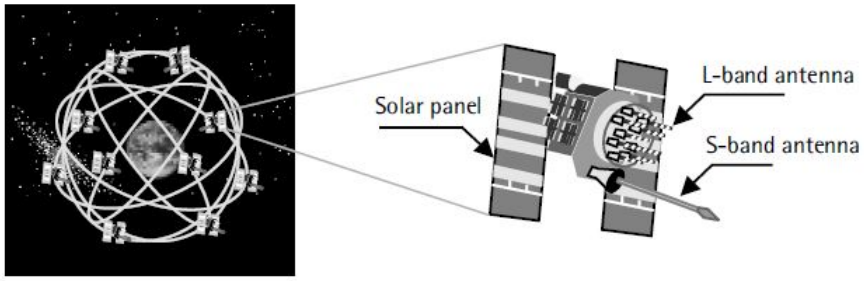
2.2 GPS

O Sistema de Posicionamento Global, sigla do inglês (GPS), foi desenvolvido pelo departamento de defesa dos EUA na década de 70 originalmente para fins militares e posteriormente tornou-se disponível para usuários civis (KLEUSBERG, 1990). Esse sistema é capaz de fornecer informações sobre posição e tempo em qualquer lugar no mundo independente das condições climáticas. Devido a questões de segurança, já que pode ser utilizado por um número ilimitado de pessoas, o GPS é um sistema passivo, ou seja, os usuários podem apenas receber os sinais de satélite (OXLEY, 2017).

O GPS é composto por uma constelação de 24 satélites operacionais. Para garantir a cobertura global os satélites são organizados de modo que quatro satélites sejam colocados

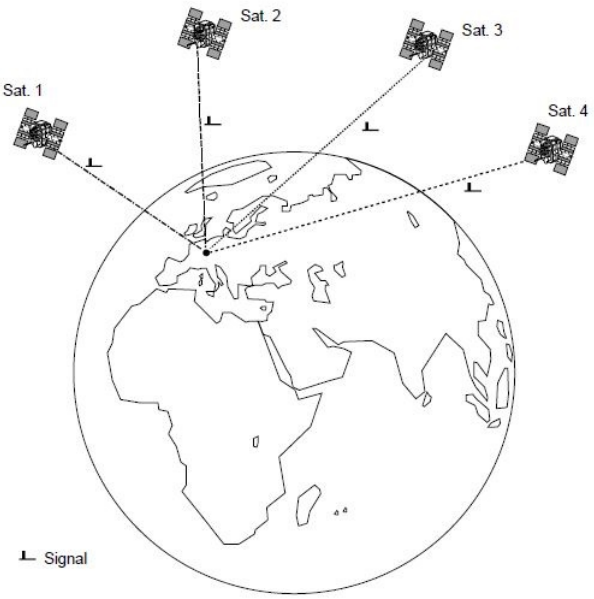
em cada um dos seis orbitais planos, fig.5. A partir dessa geometria de constelação, quatro a dez satélites de GPS serão visíveis em qualquer local do mundo desde de que seja considerado um ângulo de elevação de 10°, fig.6 (OXLEY, 2017). Segundo (OF, 2005) para a obtenção dos dados de posição ou localização apenas 4 satélites são necessários. Isso justifica, como veremos mais adiante, o fato de a shield GPS conseguir realizar a gravação de dados apenas após o reconhecimento de no mínimo 4 satélites.

Figura 5: Constelação GPS



Fonte: (OXLEY, 2017)

Figura 6: São necessários 4 satélites para determinar a posição num espaço 3D

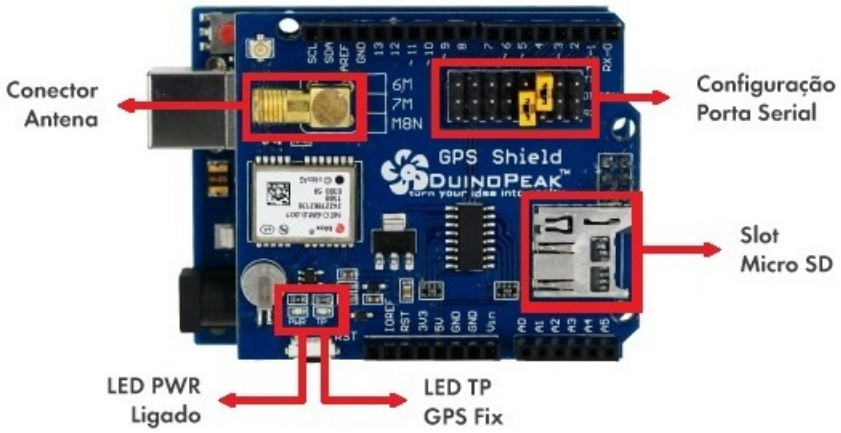


Fonte: (OF, 2005)

2.2.1 Shield GPS

A placa de GPS utilizada nesse projeto é a DuinoPeak baseada no ublox's NEO-6M que apresenta precisão de aproximadamente 2,5 metros. Essa shield é compatível com as placas UNO, Mega, Leonardo e Due. É composta por recurso de configuração de porta serial, slot de cartão micro SD, LED para indicação de PWR ligado, esse deve ascender sólido para indicar que a placa está ligada, possui também LED para indicação TP GPS Fix que deve piscar indicando que o Shield realizou a localização dos satélites e um conector para antena. A fig.7 detalha a localização de cada um dos recursos (U-BLOX, 2011) (GPS,).

Figura 7: Shield GPS



Fonte: Elaborado pela autora

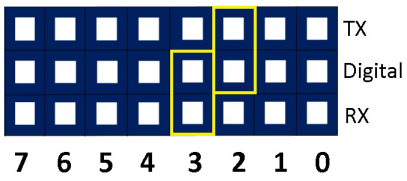
A Shield GPS possui ainda, um leitor SD que realiza a comunicação por meio da interface SPI do Arduino. Dessa forma é possível que funcione como um datalogger gravando os dados GPS em um cartão SD.

A comunicação entre placa e Arduino é feita por meio da UART, com uma velocidade de transmissão padrão de 9600bps. Podendo utilizar a interface nativa, por meio dos pinos D0 e D1, ou a interface serial via Software.

Na configuração via Software Serial com o Shield GPS é necessário utilizar jumpers nos pinos 2 e 3 conforme fig.8. No caso da configuração via Hardware, a interface serial de comunicação é selecionada entre os pinos D2 e D7 utilizando jumpers. Nesse trabalho foi considerado o primeiro caso para facilitar a programação.

É necessário ainda a utilização de uma antena GPS conforme fig.9. Essa é conectada a Shield no local indicado na fig.7.

Figura 8: Configuração Porta Serial Shield GPS.



Fonte: Elaborado pela autora

Figura 9: Antena

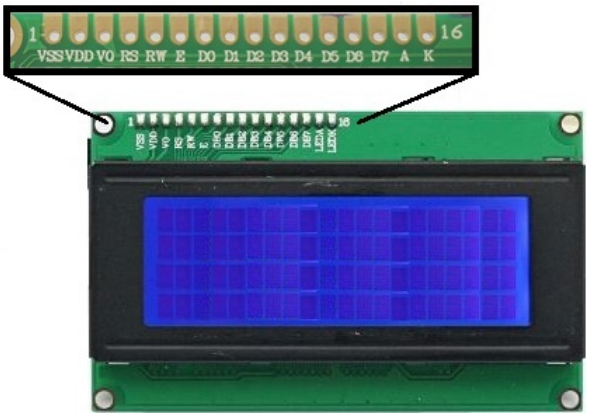
Fonte: Aliexpress

2.3 Display LCD

Um dos periféricos utilizados nesse projeto para realizar a interface com o usuário consiste em um display LCD 20x4. Esse display é um dos mais comuns encontrados no mercado. Possui ótima relação custo benefício e atende muito bem as necessidades do projeto que são principalmente simplicidade, baixo custo e robustez.

Composto por 4 linhas e 16 colunas o display possui 16 pinos para realizar sua ligação conforme pode ser observado na fig. 10 (Embedded Systems Design Laboratory, 2001).

Figura 10: Display LCD 20x4



Fonte: Elaborado pela autora

As indicações de como devem ser realizadas as ligações de cada pino são apresentadas na tab.2.

Tabela 2: Conexão entre LCD 20x4 e Arduino

Pino LCD	**Função**	**Ligação**
1	Vss	GND
2	Vdd	Vcc 5V
3	Vo	Pino central potenciômetro
4	RS	Pino 12 Arduino
5	RW	GND
6	E	Pino 11 Arduino
7	D0	Não conectado
8	D1	Não conectado
9	D2	Não conectado
10	D3	Não conectado
11	D4	Pino 5 Arduino
12	D5	Pino 4 Arduino
13	D6	Pino 3 Arduino
14	D7	Pino 2 Arduino
15	A	Vcc 5V
16	K	GND

Fonte: Elaborado pela autora

Na fig.11 é apresentada a função de cada um dos 16 pinos.

Figura 11: Pinagem do Display LCD 20x4

Pin#	Name		In/Out/Pwr
1	GND	Ground	Power
2	VCC	LCD Controller Power (+3 to +5V)	Power
3	VLCD	LCD Display Bias (+5 to -5V *see text)	Analog
4	RS	Register Select: H: Data L: Command	Input
5	R/W	H: Read L: Write	Input
6	E	Enable (Data strobe, active high)	Input
7	DB0	Data LSB	I/O
8	DB1	Data	I/O
9	DB2	Data	I/O
10	DB3	Data	I/O
11	DB4	Data	I/O
12	DB5	Data	I/O
13	DB6	Data	I/O
14	DB7	Data MSB	I/O
15	A	LED Backlight Anode (optional)	Power
16	K	LED Backlight Cathode (optional)	Power

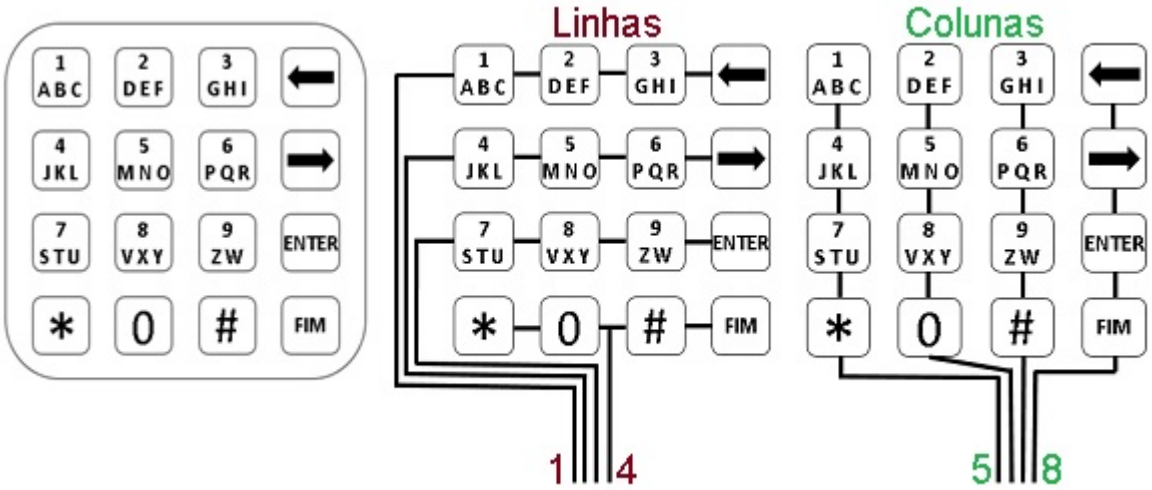
Fonte: (Embedded Systems Design Laboratory, 2001)

2.4 Teclado de Membrana

No projeto há ainda o uso de um teclado que serve para que o número de série da máquina a ser acompanhada seja inserido e o início do processo ocorra de maneira conveniente para o usuário.

O teclado escolhido foi o de membrana 4x4, muito utilizado para a entrada de dados em Arduino, esse é apresentado na fig.12. Ele é composto por 4 linhas e 4 colunas apresentando 8 pinos para a ligação com a placa. Internamente possui 16 teclas push-buttons tipo membrana, conforme podemos observar no esquemático apresentado também na fig.12. À medida que a tecla é pressionada é feita a conexão entre a linha e a coluna correspondente. Dessa forma, se for pressionada, por exemplo, a tecla de número 1 este corresponde ao pino 1 relativo a linha 1 e ao pino 5 referente a coluna 1, será feita uma conexão entre esses pinos.

Figura 12: Teclado de Membrana

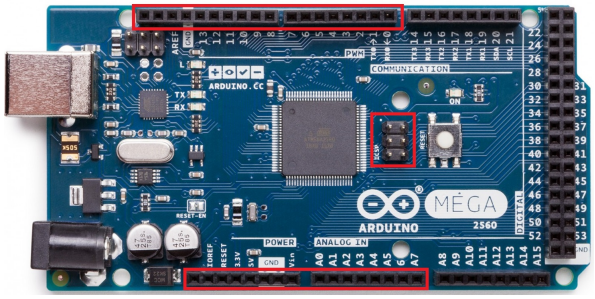


Fonte: Elaborado pela autora

2.5 Montagem do Dispositivo

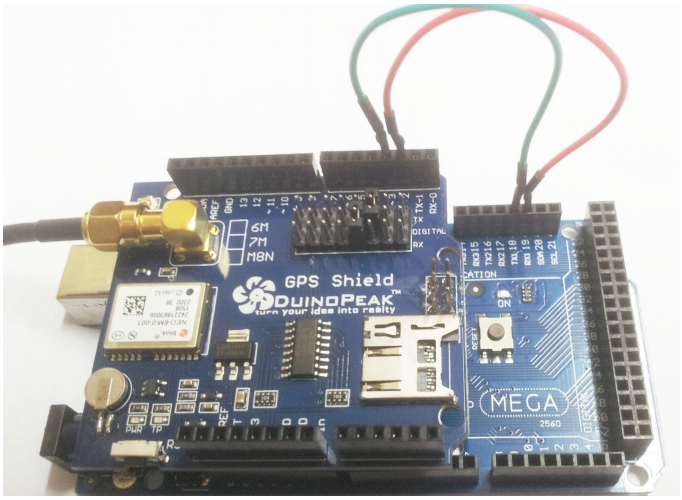
Todos os componentes citados até aqui foram utilizados para a montagem do dispositivo desenvolvido nesse projeto. Primeiramente conectou-se a Shield GPS escolhida com a placa do Arduino Mega2560 utilizando os pinos destacados na fig.13 e então realizou-se um jumper entre o pino 2 da Shield para o pino 19 do Arduino e pino 3 da Shield para o pino 18 da placa. Obtendo então a configuração conforme apresentado na fig:14

Figura 13: Pinos de ligação entre Arduino e Shield GPS



Fonte: Elaborado pela autora

Figura 14: Esquemático das ligações entre Arduino e Shield GPS

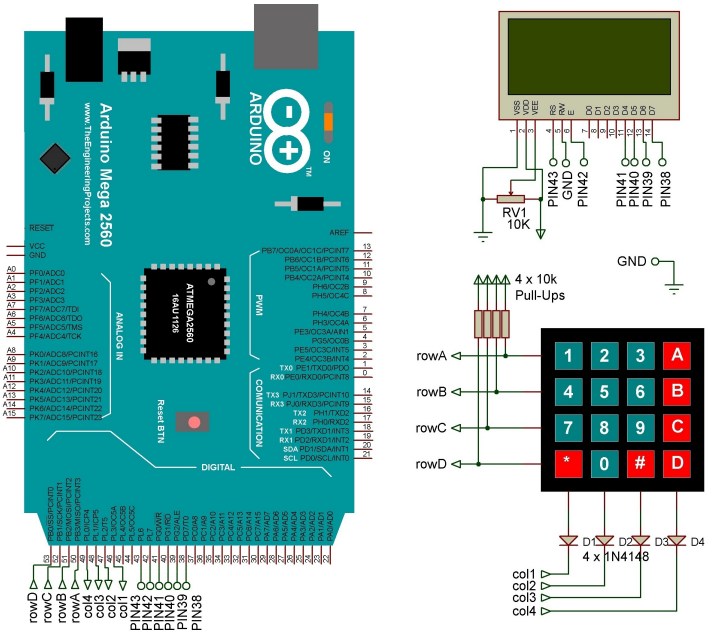


Fonte: Elaborado pela autora

Para a conexão dos periféricos de saída e entrada de dados (display LCD e teclado respectivamente), seguindo o datasheet de cada componente foi montado o circuito apresentado na fig.15.

Podemos observar que na ligação do display é bem simples. Há em adicional apenas a necessidade da inserção de um potenciômetro de 10KΩ para possibilitar o ajuste do contraste do display. Já a ligação entre teclado de membrana precisa ser conectado a 4 diodos, para evitar reflexão de sinais e proteção. Esses diodos terão uma extremidade conectada aos pinos referentes as colunas no teclado e a outra conectada aos pinos do Arduino configurados como saída. São necessários também 4 resistores de pull-up de 10KΩ conectados aos pinos referentes as linhas do teclado, essas serão conectadas ao Arduino em pinos configurados como entradas.

Figura 15: Esquemático das ligações entre Arduino, teclado e display LCD



Fonte: Elaborado pela autora

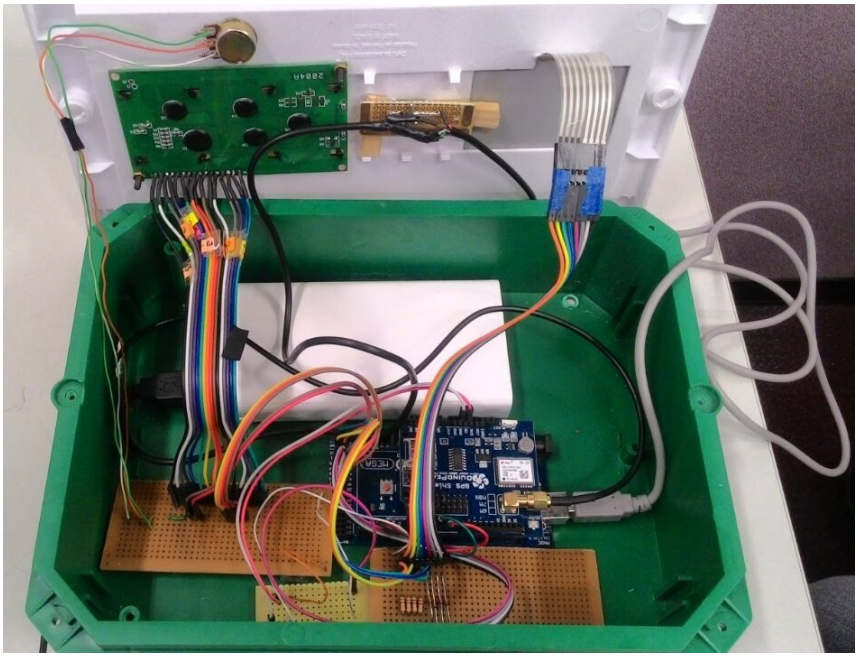
Além disso, o dispositivo precisa ser alimentado por uma fonte independente para que funcione de forma autônoma, ou seja, sem a necessidade de estar conectado a nenhum outro dispositivo. Foi então, escolhido um carregador portátil da marca Xiaomi composto por Li-Baterias de polímero com capacidade de 20000 mAh e 74Wh. Possui duas entradas USB e entradas de $5,0V/2,0A$ e $9,0V/2,0A$. É conectada por meio de um cabo USB a um botão de 3 vias e esse botão é conectado a porta serial do Arduíno por meio de outro cabo USB. Assim, a alimentação do circuito é feita utilizando a porta serial da placa. A alimentação dos periféricos é realizada pelas pinos de alimentação (5V e GND) presentes na Shield GPS que correspondem a uma extensão dos pinos do Arduino. A fig.16 ilustra o carregador portátil utilizado nesse projeto.

Figura 16: Banco de baterias utilizado para alimentação do dispositivo.

Fonte: Aliexpress

Ao final os circuitos foram colocados dentro de uma caixa adaptada para as necessidades do projeto obtendo, assim, o dispositivo apresentado na fig.17

Figura 17: Hardware do dispositivo implementado.



Fonte: Elaborado pela autora

Figura 18: Interface com o usuário.

Fonte: Elaborado pela autora

3 SOFTWARE

A implementação do software se deu na linguagem C++ e foi realizada utilizando a IDE do Arduino. A lógica do programa está apresentada na fig.19.

O arquivo gerado é do formato .CSV, que é um arquivo de valores separados por vírgula Nele é armazenada a latitude e longitude em graus, a altitude em pés, a velocidade em milhas por hora, a direção em graus, a data, o horário com fuso de Londres e o número de satélites disponíveis. A aquisição dos dados é realizada em uma taxa de 5 segundos, período que se mostrou mais seguro, durante os testes, para evitar a perda de dados relevantes.

O nome desse arquivo corresponde ao número de série da máquina a ser observada e deve conter 8 caracteres. O GPS só será carregado após a inserção desses 8 caracteres e a seleção da tecla ENTER. Assim que o usuário desejar finalizar a aquisição de dados basta desligar o dispositivo.

Para realizar a leitura dos dados armazenados é necessário abrir a caixa, retirar o cartão SD da Shield GPS e inseri-lo em um computador. O formato da saída de dados é ilustrado na fig.20.É necessário, ainda, garantir que o cartão SD está inserido antes de ligar o dispositivo. Se o cartão não for encontrado o LCD apresentará valores aleatórios e não permitirá que o número de série seja digitado.

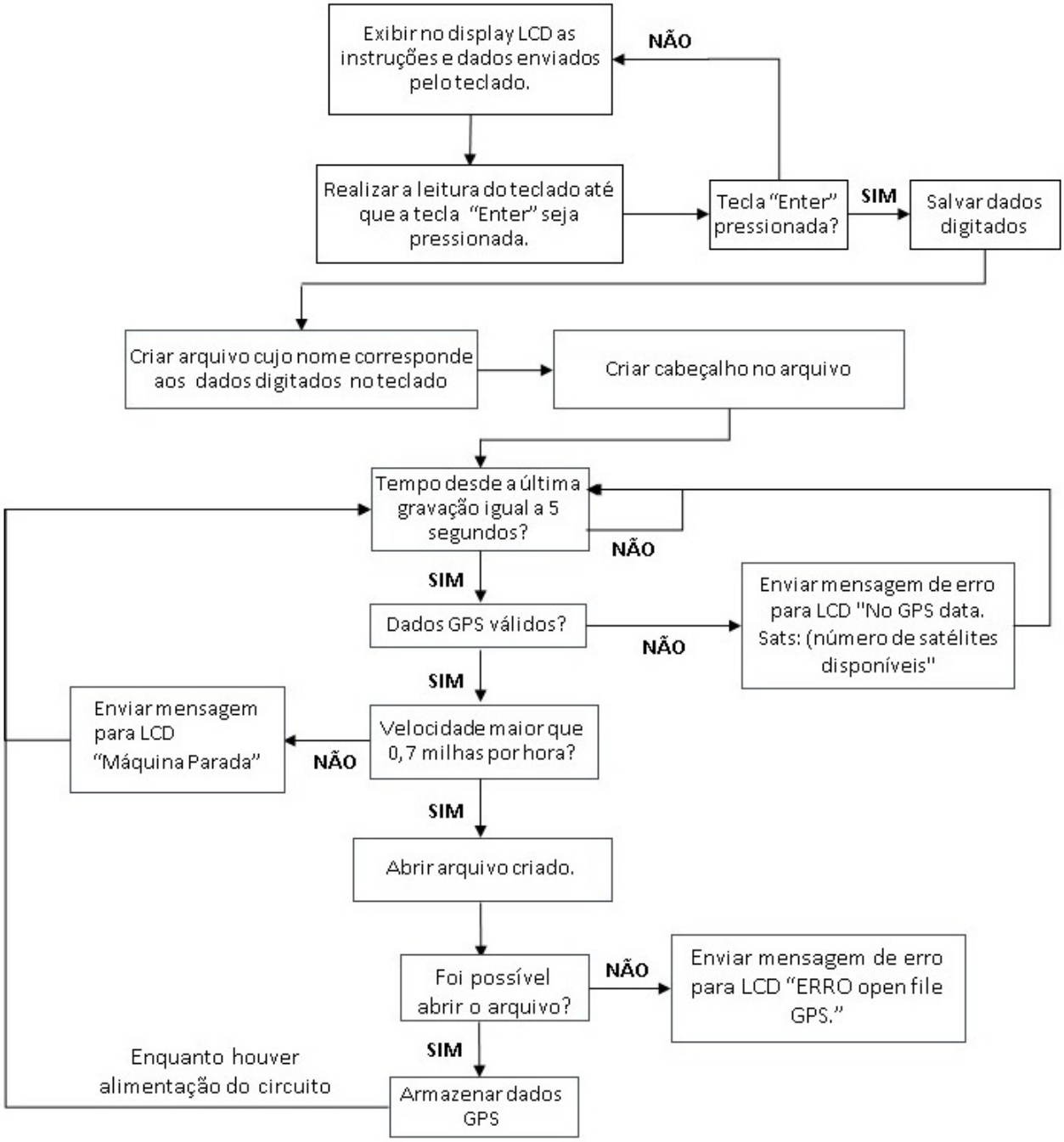
Para mais detalhes sobre o código consultar anexo.

Figura 20: Formato da saída de dados do dispositivo.

```
longitude,latitude,altitude,speed,course,date,time,satellites
-47.596794,-22.716653,1969.8,0.8,0.0,131117,10500800,7
-47.596775,-22.716661,1978.0,0.8,0.0,131117,10501500,7
-47.596702,-22.716772,2065.6,0.7,0.0,131117,10503000,7
-47.596664,-22.716814,2048.6,1.1,0.0,131117,10503500,7
-47.596660,-22.716827,2053.8,0.8,0.0,131117,10504100,7
```

Fonte: Elaborado pela autora

Figura 19: Fluxograma do programa implementado.



Fonte: Elaborado pela autora

4 TESTE, VALIDAÇÃO E RESULTADOS

Para validação do hardware e software, o dispositivo foi inserido em três máquinas motoniveladoras durante 5 dias. Pode-se constatar que após o tempo máximo de 5 dias o carregador portátil foi capaz de alimentar o circuito durante todo o teste. Além disso o equipamento suportou bem as vibrações presentes no veículo, permanecendo intacto ao final do processo.

Nos primeiros testes ao realizar o tratamento dos dados observou-se a grande presença de ruídos fazendo com que os dados de localização apresentassem imprecisões consideráveis, principalmente no período que a máquina encontrava-se em repouso. Para contornar esse problema foi inserido um condicional na programação para descartar os valores GPS quando a velocidade captada fosse menor que 0,7 milhas por hora. Com essa alteração, diminuiu-se consideravelmente a quantidade de dados irrelevantes, bem como os erros.

Pode-se constatar também, que conforme mencionado anteriormente nesse trabalho, o sinal GPS só é válido quando há no mínimo 4 satélites disponíveis. Para um valor inferior o programa continua buscando novos satélites e não realiza a coleta dos dados.

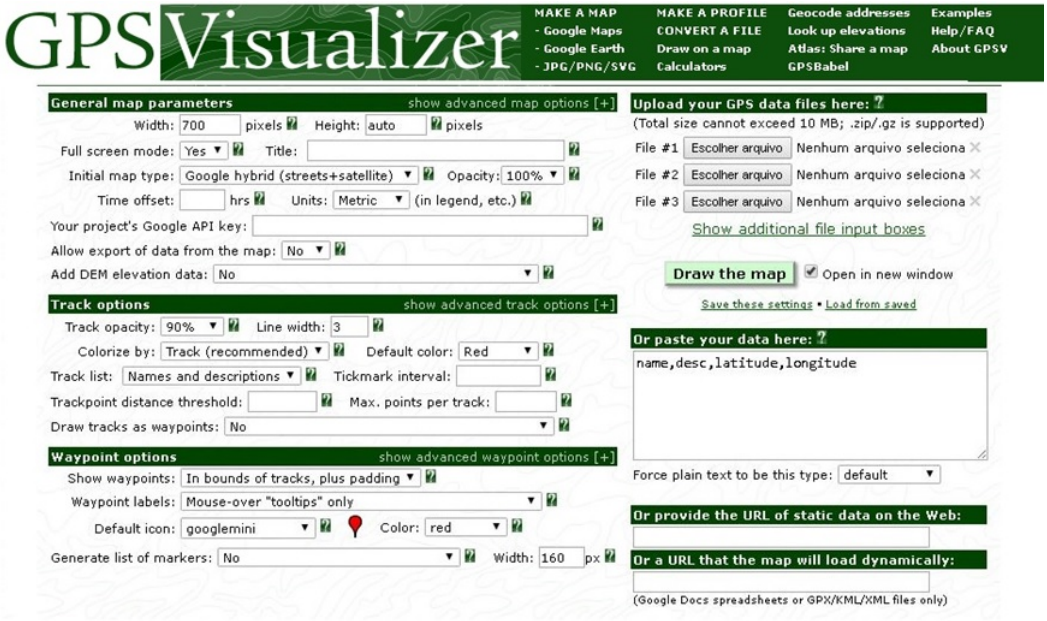
Com relação ao sinal dentro do ambiente fabril não houve problemas, já que dentro da empresa utilizada como exemplo há um replicador de sinais GPS. Assim foi possível colher uma boa amostragem de dados.

Para traçar a trajetória e distância percorrida pela máquina utilizou-se o programa "GPS Visualizer"mostrado na fig.21 disponível online. Como exemplo foi considerado um intervalo de 1 hora e plotado utilizando esse programa, obtendo, assim a fig.22. Nessa figura a linha vermelha representa a trajetória de acordo com os dados recolhidos pelo dispositivo e a linha azul é criada pelo usuário a partir da inserção de alguns pontos para determinar a distância percorrida em um possível trajeto tomando como base a linha vermelha.

É possível observar na figura que quando a máquina está parada, mesmo com o tratamento dos dados é gerado ruído que pode ser identificado pela aglomeração de tracejados vermelhos. No entanto, esse fato não representa um problema, pois como sabemos a maneira que os dados se comportam quando o veículo está parado, basta desconsiderar os respectivos dados.

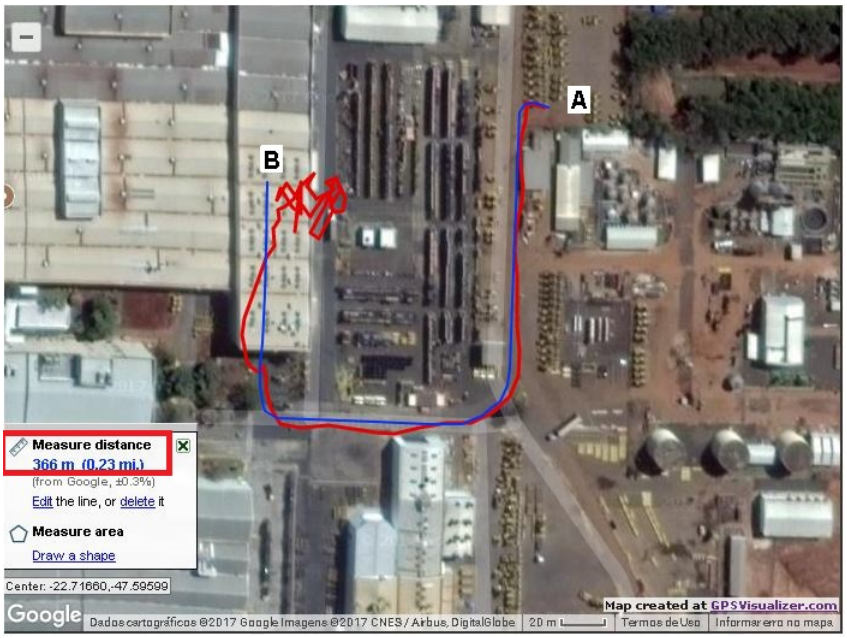
Conforme ainda a fig.22, a distância percorrida entre os pontos A e B é de 366 metros, e a partir de uma análise dos dados coletados têm-se que o tempo que a máquina fica em movimento é de 5 minutos e o tempo em repouso é de 55 minutos.

Figura 21: Software utilizado para análise de dados.



Fonte: (SCHNEIDER, 2003)

Figura 22: Trajetória realizada por uma máquina no período de 1 hora.



Fonte: Elaborado pela autora

5 CONCLUSÕES E PRÓXIMOS PASSOS

Este trabalho abordou o desenvolvimento e implementação de um dispositivo simples e de baixo custo para mapeamento da trajetória de máquinas de grande porte dentro do fluxo de testes de uma linha de produção com a finalidade de coletar dados que servirão de base para projetos de otimização dessa linha produtiva. Utilizando plataformas didáticas e comuns no mercado de embarcados foi possível desenvolver um dispositivo robusto, simples e de baixo custo, aproximadamente R$580,00, representando menos de 0,03% do valor do produto, que pode ser inserido em qualquer máquina que se queira acompanhar.

Nesse projeto além do desenvolvimento do protótipo foi realizada a validação do seu funcionamento diretamente na linha produtiva, bem como a validação da maneira de alimentação escolhida e sua autonomia.

Observou-se que é possível, utilizando esse equipamento, realizar o mapeamento da trajetória de maneira aceitável e que a tratativa dos dados representa parte extremamente importante no projeto, já que as coordenadas não são livres de erros. É essencial que os valores obtidos sejam analisados de maneira adequada para que o resultado final não distancie consideravelmente da realidade.

O próximo passo desse projeto é a implementação de uma interface com o usuário para a tratativa de dados. De forma que, qualquer pessoa, independente de sua qualificação, seja capaz de apenas escolher o arquivo que deseja tratar e o próprio programa retorne os dados de interesse, como distância total percorrida, trajetória, tempo em movimento, tempo em repouso e data das observações. Espera-se ainda em trabalhos futuros tratar a forma de acesso aos dados do cartão de memória, realizando-a de forma que não seja necessária a abertura da caixa e a remoção do SD da Shield GPS. Com isso espera-se evitar desgaste dos circuitos internos, como problemas de mau contato, desgaste do slot de cartão e do equipamento como um todo.

REFERÊNCIAS

ATMEL et al. Data Sheet Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V. **Linux-Magazin**, n. November, p. 66, 2015.

DREAMCATCHERCARRIAGE. **Some Interesting Arduino Uno R3 Projects For Beginners**. 2017. Disponível em: <https://www.dreamcatchercarriage.com/interesting-arduino-uno-r3-projects-beginners/>.

Embedded Systems Design Laboratory. HD44780 LCD Starter Guide. **Data Sheet**, p. 1–4, 2001.

GENUINO, P. **Compare board specs**. 2017. Disponível em: <https://www.arduino.cc/en/Products/Compare>.

GPS, A. Arduino GPS Shield. p. 1–4.

KLEUSBERG, A. **Comparing GPS and Glonass**. 1990. 52–54 p.

LIKER, J. K.; MEIER, D. **The Toyota way fieldbook: a practical guide for implementing Toyota's 4Ps**. [S.l.]: McGraw-Hill, 2006.

MONK, S. **Programming Arduino: getting started with sketches**. [S.l.]: McGraw Hill Professional, 2016.

OF, F. GPS Essentials of Satellite Navigation. p. 1–341, 2005.

OXLEY, A. **Introduction to GPS**. [S.l.: s.n.], 2017. 19–38 p. ISSN 1098-6596. ISBN 9788578110796.

PERFORMANCE, H. et al. Data Sheet Atmel ATmega 48P/V-88P/V-168P/V-328P. **Power**, p. 1–23, 2008.

PRODUCTS, A. **Arduino Mega 2560 Rev3**. 2017. Disponível em: <https://store.arduino.cc/usa/arduino-mega-2560-rev3>.

_____. **Arduino Uno Rev3 - Arduino Genuino**. 2017. Disponível em: <https://store.arduino.cc/arduino-uno-rev3>.

SCHNEIDER, A. **GPS Visualizer: Draw a map from a GPS data file**. 2003. Disponível em: <http://www.gpsvisualizer.com/map_input>.

SOUZA, F. **Placa Arduino MEGA 2560 - Embarcados**. 2014. Disponível em: <https://www.embarcados.com.br/arduino-mega-2560/>.

TECNOLOGIA, S. d. E. e. G. e. Aumento de Produtividade e Redução do Custo Operacional em uma Empresa de Autopeças Utilizando Conceitos Lean. **IX SEGeT**, 2012.

U-BLOX. NEO-6 GPS Modules Data Sheet. p. 25, 2011.

Anexos

ANEXO A – SOFTWARE

```
#include <TinyGPS++.h>
#include <SPI.h>
#include <SD.h>
#include <LiquidCrystal.h>
#include <Wire.h>
/*
CONFIGURACAO LCD + TECLADO
*/
//
================================================================================
============================
// --- Mapeamento de Hardware ---

//Key Pad
#define  col_1    46                //coluna 1 do teclado
#define  col_2    47                //coluna 2 do teclado
#define  col_3    48                //coluna 3 do teclado
#define  col_4    49                //coluna 4 do teclado
#define  row_A    50                //linha A  do teclado
#define  row_B    51                //linha B  do teclado
#define  row_C    52                //linha C  do teclado
#define  row_D    53                //linha D  do teclado

//DIPLAY LCD
//Criando um objeto da classe LiquidCrystal e
//inicializando com os pinos da interface.
const int rs = 43, en = 42, d4 = 41, d5 = 40, d6 = 39, d7 = 38;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//
================================================================================
============================
// --- Prototipo das Funcoes ---
void numero(int aux2, int k);            //Funcoes para imprimir o numero digitado na tela do LCD
int FuncCounter(int counter2, int indice);         //Funcao de contagem de teclas
char NextPos(char valor);

//
================================================================================
============================
// -- - Variaveis Globais--
char control = 0x01;                //variavel de controle de teclado
char counter = 0x00;                 //variavel auxiliar de contagem
int  counter2 = 0, c = 0;
int  aux = 0, aux2 = 0;
int  number = 0;                //variavel para armazenar o número pressionado no teclado
int  num[16] = { 0 };
int  auxnum[16] = { 0 };
int  k, ind = 0, t = 0, Avanc = 0;
unsigned long auxAnterior = 0, auxAtual = 0, deltaT = 0;
unsigned long inicio = 0, inicio2 = 0, agora = 0, tempo = 0, inicio1[16] = { 0 };
char str[8] = { -1 }, valor = 0, valorAnterior = 0;
bool hitEnter = false;
int setaLeft = 0;
```

```
//
================================================================================
============================
//// --- Interrupcao ---
ISR(TIMER2_OVF_vect)
{
	TCNT2 = 2000;		// Reinicializa o registrador do Timer2
	counter++;		// incrementa counter

	if (counter == 0x05) // counter igual a D'5'?
	{ // sim...
		counter = 0x00;  //reinicia counter

	} //end if counter
}

void readFileNameKeyboard() {

	while (hitEnter == false) {
		if (digitalRead(col_1) && control == 0x01)	//Coluna 1 em nível high? Control igual 1?
		{ //Sim...
			control = 0x02;			//control igual a 2
			digitalWrite(col_1, LOW);		//apenas coluna 1 em nivel baixo
			digitalWrite(col_2, HIGH);
			digitalWrite(col_3, HIGH);
			digitalWrite(col_4, HIGH);

			// -- Testa qual tecla foi pressionada na coluna 1 e armazena o valor --
			if (!digitalRead(row_A)) {
				k = 1;   //Posicaoo do número 1 no vetor
				num[k] = 1; //vetor posicao atribuindo "verdadeiro" a possicaoo 1
				c++;

				for (int i = 0; i < 17; i++) {
					if (i != k) {
						num[i] = 0;
					}
				}
				counter2 = 1;
				FuncCounter(counter2, k);

			}// end if row_A

			else if (!digitalRead(row_B)) {
				k = 4;			//Posicao do numero 4 no vetor
				num[k] = 1;			//vetor posicao atribuindo "verdadeiro" a posicao 4
				c++;

				for (int i = 0; i < 17; i++) {
					if (i != k) {
						num[i] = 0;
					}
				}
				counter2 = 1;
				FuncCounter(counter2, k);
			}//end if row_B
```

```
        else if (!digitalRead(row_C)) {
                k = 7;                          //Posicaoo do numero 7 no vetor
                num[k] = 1;                       //vetor posicao atribuindo "verdadeiro" a posicao 7
                c++;

                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);
        }//end if row_C

        else if (!digitalRead(row_D)) {
                k = 10;                          //Posicao do numero 3 no vetor
                c++;
                num[k] = 1;                     //vetor pos icao o atribuindo "verdadeiro" a posicao1
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 0;
                FuncCounter(counter2, k);
        }//end if row_D

}//end if col_1

else if (digitalRead(col_2) && control == 0x02)  //Coluna 2 em nivel high? Control igual 2?
{ //Sim...
        control = 0x03;                         //control igual a 3
        digitalWrite(col_1, HIGH);
        digitalWrite(col_2, LOW);               //apenas coluna 2 em nivel baixo
        digitalWrite(col_3, HIGH);
        digitalWrite(col_4, HIGH);

        // -- Testa qual tecla foi pressionada na coluna 2 e armazena o valor --
        if (!digitalRead(row_A)) {
                k = 2;                          //Posicao o do numero 2 no vetor
                num[k] = 1;                    //vetor posicao o atribuindo "verdadeiro" a posição o 2
                c++;
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);

        }// end if row_A da coluna 2

        else if (!digitalRead(row_B)) {
                k = 5;                          // posição o do numero 5 no vetor
                c++;
```

```
                num[k] = 1;     //vetor posição o atribuindo "verdadeiro" a posição o 5
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);
        }//end if row_B

        else if (!digitalRead(row_C)) {
                k = 8;              // posição  do numero 8 no vetor
                c++;
                num[k] = 1;   //vetor posição o atribuindo "verdadeiro" a  posição o 8
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);

        }//end if row_C

        else if (!digitalRead(row_D)) {
                k = 0;                          // posição do numero 3 no vetor
                c++;
                num[k] = 1;               //vetor posição  atribuindo "verdadeiro" a posição o 3
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 0;
                FuncCounter(counter2, k);
        }//end if row_D

} //end if col_2

else if (digitalRead(col_3) && control == 0x03)  //Coluna 3 em nivel high? Control igual 3?
{ //Sim...
        control = 0x04;                         //control igual a 4
        digitalWrite(col_1, HIGH);                //
        digitalWrite(col_2, HIGH);
        digitalWrite(col_3, LOW);                //apenas coluna 3 em nivel baixo
        digitalWrite(col_4, HIGH);

        // -- Testa qual tecla foi pressionada na coluna 3 e armazena o valor --
        if (!digitalRead(row_A)) {
                k = 3;                          // posição o do numero 3 no vetor
                c++;
                num[k] = 1;               //vetor posição o atribuindo "verdadeiro" a posição o 3
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
```

```
                }
                counter2 = 1;
                FuncCounter(counter2, k);

        }// end if row_A da coluna 3

        else if (!digitalRead(row_B)) {
                k = 6;                  // posição o do numero 6 no vetor
                c++;
                num[k] = 1;          //vetor posição atribuindo "verdadeiro" a posição o 6
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);

        }//end if row_B

        else if (!digitalRead(row_C)) {
                k = 9;                          // posição o do numero 9 no vetor
                c++;
                num[k] = 1;              //vetor posição  atribuindo "verdadeiro" a posição o 9
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 1;
                FuncCounter(counter2, k);

        }//end if row_C

        else if (!digitalRead(row_D)) {
                k = 11;                          // posição do numero '0' no vetor
                c++;
                num[k] = 1;               //vetor posição o atribuindo "verdadeiro" a posição o '11'
                for (int i = 0; i < 17; i++) {
                        if (i != k) {
                                num[i] = 0;
                        }
                }
                counter2 = 0;
                FuncCounter(counter2, k);
        }//end if row_D

} //end if col_3

else if (digitalRead(col_4) && control == 0x04)  //Coluna 3 em nível high? Control igual 4?
{ //Sim...
        control = 0x01;                         //control igual a 1
        digitalWrite(col_1, HIGH);               //
        digitalWrite(col_2, HIGH);
        digitalWrite(col_3, HIGH);
        digitalWrite(col_4, LOW);               //apenas coluna 4 em nivel baixo
```

```
                                                                                    // -- Testa
qual tecla foi pressionada e armazena o valor --
                        if (!digitalRead(row_A)) { // Seta GoRight
                                k = 12;                         // posição o do numero 3 no vetor
                                c++;
                                num[k] = 1;           //vetor posição o atribuindo "verdadeiro" posição o 12
                                for (int i = 0; i < 17; i++) {
                                        if (i != k) {
                                                num[i] = 0;
                                        }
                                }
                                counter2 = 1;
                                FuncCounter(counter2, k);
                        }
                        else if (!digitalRead(row_B)) { // Seta GoLeft
                                k = 13;
                                c++;
                                num[k] = 1;             //vetor posição atribuindo "verdadeiro" a posição 13
                                for (int i = 0; i < 17; i++) {
                                        if (i != k) {
                                                num[i] = 0;
                                        }
                                }
                                counter2 = 1;
                                FuncCounter(counter2, k);
                        }
                        else if (!digitalRead(row_C)) { // Enter
                                hitEnter = true;
                                //  lcd.cursor();
                                //  lcd.setCursor(0, 2);
                                // lcd.print("Conectando GPS...  ");
                                //  delay(600);
                        }
                        else if (!digitalRead(row_D)) { //Fim

                        }

                } //end if col_4
        }
}


char NextPos(char valor) {

        if (t == 0 && str[0] == -1) {
                str[t] = valor;
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                auxAnterior = millis();
                delay(250);
        }
```

```
        else if (t < 7 && str[0] != -1) {
                t = t + 1;
                str[t] = valor;
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                auxAnterior = millis();
        }
        if (t == 7) {
                lcd.cursor();
                lcd.setCursor(0, 2);
                lcd.print("ENTER P/Confirmar  ");
        }
        //  for (int b = 0; b <= t; b++) {
        //    Serial.println(str[b]);
        //  }
}

int FuncCounter(int counter2, int k)  //Funcao de contagem de cada tecla
{
        TIMSK2 = 0x00;                  //Desabilita interrupcao do Timer2
        delay(350);

        inicio2 = inicio1[k]; // variavel que armazena o momento que a tecla foi pressionada pela  ultima
vez, alterada na funcao numero
        agora = millis();
        tempo = agora - inicio2;

        if (tempo > 1200) auxnum[k] = 0; // se o tempo e maior que 190 zera a variavel que detecta quantas
vezes a tecla foi pressionada

        auxnum[k] = counter2 + auxnum[k];

        if (k < 10) {
                if (auxnum[k] > 4) auxnum[k] = 1;
        }

        aux2 = auxnum[k];
        numero(aux2, k);
        TIMSK2 = 0x01;   //Habilita interrupcao do Timer2

}//end funcao de contagem

 //int enter();
 //Aparecer mensagem "Adiquirindo localização o GPS, ao final do processo precione fim para finalizar"


void numero(int aux2, int k)                    //Funcao para imprimir o numero digitado
{

        if (num[1]) {
                inicio1[1] = millis();

                if (t == 0 && c == 1) Avanc = k;
```

```
        if (aux2 == 1) valor = '1';
        else if (aux2 == 2) valor = 'A';
        else if (aux2 == 3) valor = 'B';
        else if (aux2 == 4) valor = 'C';
        auxAtual = millis();
        deltaT = auxAtual - auxAnterior;

        if (Avanc == k && deltaT < 1200) {
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                str[t] = valor;
        }
        else NextPos(valor);

        Avanc = 1;
        auxAnterior = millis();

}//fim if numero 1

else if (num[2]) {
        inicio1[2] = millis();

        if (t == 0 && c == 1) Avanc = k;

        if (aux2 == 1) valor = '2';
        else if (aux2 == 2) valor = 'D';
        else if (aux2 == 3) valor = 'E';
        else if (aux2 == 4) valor = 'F';
        auxAtual = millis();
        deltaT = auxAtual - auxAnterior;

        if (Avanc == k && deltaT < 1200) {
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                str[t] = valor;
        }
        else NextPos(valor);

        Avanc = 2;
        auxAnterior = millis();

}//fim if numero 2

else if (num[3]) {
        inicio1[3] = millis();

        if (t == 0 && c == 1) Avanc = k;

        if (aux2 == 1) valor = '3';
        else if (aux2 == 2) valor = 'G';
        else if (aux2 == 3) valor = 'H';
        else if (aux2 == 4) valor = 'I';
```

```
		auxAtual = millis();
		deltaT = auxAtual - auxAnterior;

		if (Avanc == k && deltaT < 1200) {
			lcd.cursor();
			lcd.setCursor(t, 1);
			lcd.print(valor);
			lcd.setCursor(t, 1);
			str[t] = valor;
		}
		else NextPos(valor);

		Avanc = 3;
		auxAnterior = millis();
}//fim if numero 3

else if (num[4]) {
		inicio1[4] = millis();

		if (t == 0 && c == 1) Avanc = k;

		if (aux2 == 1) valor = '4';
		else if (aux2 == 2) valor = 'J';
		else if (aux2 == 3) valor = 'K';
		else if (aux2 == 4) valor = 'L';
		auxAtual = millis();
		deltaT = auxAtual - auxAnterior;

		if (Avanc == k && deltaT < 1200) {
			lcd.cursor();
			lcd.setCursor(t, 1);
			lcd.print(valor);
			lcd.setCursor(t, 1);
			str[t] = valor;
		}
		else NextPos(valor);

		Avanc = 4;
		auxAnterior = millis();
}// final if numero 4

else if (num[5]) {
		inicio1[5] = millis();

		if (t == 0 && c == 1) Avanc = k;

		if (aux2 == 1) valor = '5';
		else if (aux2 == 2) valor = 'M';
		else if (aux2 == 3) valor = 'N';
		else if (aux2 == 4) valor = 'O';
		auxAtual = millis();
		deltaT = auxAtual - auxAnterior;

		if (Avanc == k && deltaT < 1200) {
			lcd.cursor();
			lcd.setCursor(t, 1);
```

```
                lcd.print(valor);
                lcd.setCursor(t, 1);
                str[t] = valor;
        }
        else NextPos(valor);

        Avanc = 5;
        auxAnterior = millis();
}// final if numero 5

else if (num[6]) {
        inicio1[6] = millis();
        if (t == 0 && c == 1) Avanc = k;

        if (aux2 == 1) valor = '6';
        else if (aux2 == 2) valor = 'P';
        else if (aux2 == 3) valor = 'Q';
        else if (aux2 == 4) valor = 'R';
        auxAtual = millis();
        deltaT = auxAtual - auxAnterior;

        if (Avanc == k && deltaT < 1200) {
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                str[t] = valor;
        }
        else NextPos(valor);

        Avanc = 6;
        auxAnterior = millis();
}//final if numero 6

else if (num[7]) {
        inicio1[7] = millis();
        if (t == 0 && c == 1) Avanc = k;

        if (aux2 == 1) valor = '7';
        else if (aux2 == 2) valor = 'S';
        else if (aux2 == 3) valor = 'T';
        else if (aux2 == 4) valor = 'U';
        auxAtual = millis();
        deltaT = auxAtual - auxAnterior;

        if (Avanc == k && deltaT < 1200) {
                lcd.cursor();
                lcd.setCursor(t, 1);
                lcd.print(valor);
                lcd.setCursor(t, 1);
                str[t] = valor;
        }
        else NextPos(valor);

        Avanc = 7;
        auxAnterior = millis();
```

```
}//final if numero 7

else if (num[8]) {
	inicio1[8] = millis();
	if (t == 0 && c == 1) Avanc = k;

	if (aux2 == 1) valor = '8';
	else if (aux2 == 2) valor = 'V';
	else if (aux2 == 3) valor = 'X';
	else if (aux2 == 4) valor = 'Z';
	auxAtual = millis();
	deltaT = auxAtual - auxAnterior;

	if (Avanc == k && deltaT < 1200) {
		lcd.cursor();
		lcd.setCursor(t, 1);
		lcd.print(valor);
		lcd.setCursor(t, 1);
		str[t] = valor;
	}
	else NextPos(valor);

	Avanc = 8;
	auxAnterior = millis();
}//final if numero 8

else if (num[9]) {
	inicio1[9] = millis();
	if (t == 0 && c == 1) Avanc = k;

	if (aux2 == 1) valor = '9';
	else if (aux2 == 2) valor = 'W';
	else if (aux2 == 3) valor = 'Y';
	else if (aux2 == 4) valor = ' ';
	auxAtual = millis();
	deltaT = auxAtual - auxAnterior;

	if (Avanc == k && deltaT < 1200) {
		lcd.cursor();
		lcd.setCursor(t, 1);
		lcd.print(valor);
		lcd.setCursor(t, 1);
		str[t] = valor;
	}
	else NextPos(valor);

	Avanc = 9;
	auxAnterior = millis();
}//final if numero 9

else if (num[0]) {
	inicio1[0] = millis();
	if (t == 0 && c == 1) Avanc = k;

	valor = '0';
	auxAtual = millis();
```

```
		deltaT = auxAtual - auxAnterior;

		NextPos(valor);

		Avanc = 0;
		auxAnterior = millis();
	}

	else if (num[10]) {
		inicio1[10] = millis();

		valor = '*';
		auxAtual = millis();
		deltaT = auxAtual - auxAnterior;

		NextPos(valor);

		Avanc = 10;
		auxAnterior = millis();
	}
	else if (num[11]) {
		inicio1[11] = millis();
		if (t == 0 && c == 1) Avanc = k;

		valor = '#';
		auxAtual = millis();
		deltaT = auxAtual - auxAnterior;

		NextPos(valor);

		Avanc = 11;
		auxAnterior = millis();
	}
	else if (num[12]) { //Go Left
		t = t - 1;
		if (t < 0) t = -1;
		lcd.cursor();
		lcd.setCursor(t + 1, 1);
		setaLeft = 1;

	}
	else if (num[13]) { //Go Right
		if (setaLeft == 1) {
			if (t > 7) t = 7;
				lcd.cursor();
				lcd.setCursor(t + 1, 1);
				setaLeft = 0;
		}
		else {
			t = t + 1;
			if (t > 7) t = 7;
			lcd.cursor();
			lcd.setCursor(t + 1, 1);
		}
	}
}//end numero
```

```
 /*
CONFIGURACAO  GPS

 */

#define ARDUINO_USD_CS 8 // uSD card CS pin (pin 8 on Duinopeak GPS Logger Shield)

 ////////////////////////
 // Log File Defintions //
 ////////////////////////
 // Keep in mind, the SD library has max file name lengths of 8.3 - 8 char prefix,
 // and a 3 char suffix.
#define LOG_FILE_SUFFIX "csv" // Suffix of the log file
char logFileName[13]; // Char string to store the log file name

// Data to be logged:
#define LOG_COLUMN_COUNT 8

char * log_col_names[LOG_COLUMN_COUNT] = {
        "longitude", "latitude", "altitude", "speed", "course", "date", "time", "satellites"
}; // log_col_names is printed at the top of the file.

   ///////////////////////
   // Log Rate Control //
   ///////////////////////
#define LOG_RATE 5000 // Log every 5 seconds
unsigned long lastLog = 0; // Global var to keep of last time we logged

 ////////////////////////
// TinyGPS Definitions //
////////////////////////

TinyGPSPlus tinyGPS; // tinyGPSPlus object to be used throughout
#define GPS_BAUD 9600 // GPS module's default baud rate

//////////////////////////////////
// GPS Serial Port Definitions //
//////////////////////////////////

#define ARDUINO_GPS_RX 3 // GPS TX, Arduino RX pin
#define ARDUINO_GPS_TX 2 // GPS RX, Arduino TX pin

// Define the serial monitor port. On the Uno, Mega, and Leonardo this is 'Serial'
//  on other boards this may be 'SerialUSB'
#define SerialMonitor Serial

void setup()
{
        lcd.begin(20, 4);
        lcd.clear();
        lcd.cursor();
        lcd.setCursor(0, 0);
        lcd.print("Digite Num.de Serie:");
        lcd.setCursor(0, 3);
        lcd.print("Pres.Fim p/Encerrar");
```

```
	Serial.begin(9600);

	for (char i = 46; i < 50; i++) pinMode(i, OUTPUT);     //Saídas para varredura das colunas
	for (char j = 50; j < 54; j++) pinMode(j, INPUT);      //Entradas para as linhas

	digitalWrite(col_1, HIGH);                    //Inicializa coluna 1 em HIGH
	digitalWrite(col_2, HIGH);                    //Inicializa coluna 2 em HIGH
	digitalWrite(col_3, HIGH);                    //Inicializa colune 3 em HIGH
	digitalWrite(col_4, HIGH);                    //Inicializa colune 3 em HIGH

	TCCR2A = 0x00;   //Timer operando em modo normal
	TCCR2B = 0x07;   //Prescaler 1:1024
	TCNT2 = 2000;    //0,5 s overflow again
	TIMSK2 = 0x01;   //Habilita interrupcao do Timer2

	readFileNameKeyboard();

	Serial1.begin(GPS_BAUD);

	lcd.cursor();
	lcd.setCursor(0, 2);
	lcd.print("Setting up SD card.");
	SerialMonitor.println("Setting up SD card.");
	// see if the card is present and can be initialized:
	if (!SD.begin(ARDUINO_USD_CS))
	{
		SerialMonitor.println("Error initializing SD card.");
		lcd.cursor();
		lcd.setCursor(0, 2);
		lcd.print("ERRO SD card.      ");
	}
	updateFileName(); // Each time we start, create a new file, increment the number
	printHeader(); // Print a header at the top of the new file
}

void loop()
{
	if ((lastLog + LOG_RATE) <= millis())
	{ // If it's been LOG_RATE milliseconds since the last log:
		if (tinyGPS.location.isUpdated()) // If the GPS data is vaild
		{
			if (tinyGPS.speed.mph() >= 0.7)
			{
				if (logGPSData()) // Log the GPS data
				{
					lcd.cursor();
					lcd.setCursor(0, 2);
					lcd.print("Gravando Trajetoria.");
					//SerialMonitor.println("GPS logged."); // Print a debug message
					lastLog = millis(); // Update the lastLog variable
				}
				else // If we failed to log GPS
				{ // Print an error, don't update lastLog
					lcd.cursor();
					lcd.setCursor(0, 2);
```

```
                                        lcd.print("ERRO open file GPS. ");
                                        //SerialMonitor.println("Failed to log new GPS data.");
                                }
                        }
                        else {
                                lcd.cursor();
                                lcd.setCursor(0, 2);
                                lcd.print("   Maquina Parada.  ");
                        }
                }
                else // If GPS data isn't valid
                {
                        // Print a debug message. Maybe we don't have enough satellites yet.
                        lcd.cursor();
                        lcd.setCursor(0, 2);
                        lcd.print("No GPS data.Sats:");
                        lcd.setCursor(17, 2);
                        lcd.print(tinyGPS.satellites.value());
                        //    SerialMonitor.print("No GPS data. Sats: ");
                        //    SerialMonitor.println(tinyGPS.satellites.value());
                }
        }

        // If we're not logging, continue to "feed" the tinyGPS object:
        while (Serial1.available())
                tinyGPS.encode(Serial1.read());
}

byte logGPSData()
{
        File logFile = SD.open(logFileName, FILE_WRITE); // Open the log file

        if (logFile)
        {
                // Print longitude, latitude, altitude (in feet), speed (in mph), course
                // in (degrees), date, time, and number of satellites.
                logFile.print(tinyGPS.location.lng(), 6);
                logFile.print(',');
                logFile.print(tinyGPS.location.lat(), 6);
                logFile.print(',');
                logFile.print(tinyGPS.altitude.feet(), 1);
                logFile.print(',');
                logFile.print(tinyGPS.speed.mph(), 1);
                logFile.print(',');
                logFile.print(tinyGPS.course.deg(), 1);
                logFile.print(',');
                logFile.print(tinyGPS.date.value());
                logFile.print(',');
                logFile.print(tinyGPS.time.value());
                logFile.print(',');
                logFile.print(tinyGPS.satellites.value());
                logFile.println();
                logFile.close();
                // Return successa
                return 1;
        }
```

```
	return 0; // If we failed to open the file, return fail
}

// printHeader() - prints our eight column names to the top of our log file
void printHeader()
{
	File logFile = SD.open(logFileName, FILE_WRITE); // Open the log file

	if (logFile) // If the log file opened, print our column names to the file
	{
		int i = 0;
		for (; i < LOG_COLUMN_COUNT; i++)
		{
			logFile.print(log_col_names[i]);
			if (i < LOG_COLUMN_COUNT - 1) // If it's anything but the last column
				logFile.print(','); // print a comma
			else // If it's the last column
				logFile.println(); // print a new line
		}
		logFile.close(); // close the file
	}
}

// updateFileName() - Looks through the log files already present on a card,
// and creates a new file with an incremented file index.
void updateFileName()
{

	memset(logFileName, 0, strlen(logFileName)); // Clear logFileName string
	for (int i = 0; i < 9; i++) {
		if (str[i] == -1) {
			str[i] = '\0';
		}
		if (i == 8) {
			str[8] = '\0';
		}
	}
	sprintf(logFileName, "%s.%s", str, LOG_FILE_SUFFIX);

	if (SD.exists(logFileName)) // If a file doesn't exist
	{
		SerialMonitor.print(logFileName);
		SerialMonitor.println(" exists"); // Print a debug statement
	}

	SerialMonitor.print("File name: ");
	SerialMonitor.println(logFileName); // Debug print the file name
}
```