UNIVERSIDADE DO VALE DO PARAÍBA
FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO
ENGENHARIA ELÉTRICA / ELETRÔNICA

BRAÇO ARTICULADO COM CONTROLE PROPORCIONAL DE MOVIMENTO COMANDADO VIA BLUETOOTH POR UM APLICATIVO DESENVOLVIDO PARA PLATAFORMAS ANDROID

ERICK DA PENHA FERREIRA
NICHOLAS DE LACERDA ALCIDES ALVES

Orientador: Alessandro Correa Mendes

São José dos Campos - SP
Novembro de 2013

RESUMO

O projeto em questão se trata de um braço robótico controlado remotamente via celular com comandos seriais enviados por interface Bluetooth. Esse braço é comandado por seis servo motores com controle proporcional de movimento através de um aplicativo para dispositivos com plataforma Android, aplicativo esse elaborado pelos próprios alunos responsáveis pelo projeto. Nesse relatório será abordado alguns tópicos como o kit de desenvolvimento *LaunchPad* MSP430, interface Bluetooth, programação de aplicativos para plataforma Android, servo motores e controle proporcional dos mesmos. Este braço robótico foi dividido em três partes. A estrutura mecânica foi feita a partir de alumínio com suas peças elaboradas no AutoCAD que, acopladas a seis servo motores, possibilitam o movimento do braço. Um aplicativo para dispositivos com sistema operacional Android foi desenvolvido para enviar, via Bluetooth, um pacote de números de 0 a 180 referentes ao ângulo de rotação de cada servo. Estes números são controlados através de uma *slide bar* onde permite ao usuário um controle proporcional de movimento. O microcontrolador MSP430 com o auxílio de um módulo Bluetooth recebe estes dados do aplicativo e posteriormente envia um sinal PWM a cada servo posicionando seus eixos no ângulo desejado. Devido ao consumo dos servos, uma fonte de alimentação externa foi inserida no sistema para suprir a energia exigida. Este projeto tem por objetivo auxiliar os estudantes de engenharia a ter um maior contato com a robótica e funcionará como uma ferramenta de ensino e aplicação dos conhecimentos adquiridos em sala, unindo os conhecimentos obtidos nas matérias de Microcontroladores e Microprocessadores, Lógica para programação, Sistemas Robóticos, Projeto em Engenharia, Eletrônica analógica e Digital, entre outras.

Palavras-chave: Robótica, *LaunchPad*, Bluetooth, Android, Servo motor.

ABSTRACT

This technical report describes the development process of the robotic arm, wirelessly controlled by Bluetooth interface. The device runs on the Android Operational System and an Application controls the six servo motors responsible for the arm movement. On next chapters it will be discussed topics related to development of this project, such as the LaunchPad MSP430 development kit, Bluetooth interface, application programming, servo motors and its proportional moving control. This robotic arm was divided into three parts. The mechanical structure is made from aluminum with their pieces developed in AutoCAD, six servo motors, were coupled to allow the arm movement. An application for devices with Android operating system was developed to send, via bluetooth, a package of numbers from 0 to 180 in reference to the rotation angle of each servo. These numbers are controlled through a slide bar which allows the user a proportional motion control. The MSP430 microcontroller working together a bluetooth module receives the data from the application and then sends a PWM signal to each servo positioning their axes at the desired angle. Due to consumption of the servo motors, one external power supply was inserted in the system to supply the energy required. The main purpose of this project is to assist Engineering students to test their knowledge obtained during the course in respect of robotic systems, programming and others related subjects.

Keywords: Robotic, LaunchPad, Bluetooth, Android, Servo motor.

INTRODUÇÃO

Um robô é definido como um manipulador multipropósito controlado manualmente ou automaticamente e pode ser programável em um ou mais eixos[1]. Usualmente o termo robótica emprega-se para indicar a disciplina associada ao uso da programação de robôs e a expressão engenharia robótica é mais específica e refere-se à construção de robôs e dispositivos robóticos [2].

Um dos equipamentos mais utilizados dentro de fábricas e montadoras são os robôs. Eles são responsáveis por diversos trabalhos, onde antes eram executados somente por seres humanos. Uma grande vantagem dos robôs é que podem trabalhar por horas sem a necessidade de parar para descansar, além de serem muito precisos uma vez programados corretamente. Outra vantagem dos robôs é a possibilidade de trabalharem em áreas de riscos, preservando o ser humano.

Nesse trabalho foi desenvolvido um braço robótico, ou também conhecido como manipulador robótico, que é comandado através de tecnologias de fácil acesso ao público e que estão intensamente presentes no dia a dia das pessoas, como a comunicação Bluetooth e celular com plataforma Android, além de utilizar servo motores que são largamente aplicados em robótica e modelismo. Outro fator interessante que será mostrado é o desenvolvimento de um aplicativo para celular dedicado a essa função, aplicativo esse completamente elaborado pelos responsáveis desse projeto, produzido no programa chamado *App Inventor*.

A utilização do kit de desenvolvimento da *Texas Instruments*, o *LaunchPad* MSP430 trabalhando com valores analógicos, PWM e comunicação Bluetooth é outro elemento que merece destaque, pois esse controlador está cada vez mais difundido na área de programação e está funcionando com outras tecnologias vastamente empregadas.

O objetivo desse projeto é o desenvolvimento de um braço robótico com seis eixos de movimentação incluindo uma garra em sua extremidade. Controlado via Bluetooth por um aplicativo para dispositivos com Sistema Operacional Android, este deve possibilitar o controle proporcional de movimento dos servos através de uma *slide bar* na tela do aplicativo. O braço deve ser projetado, desde a estrutura mecânica até os servo motores, de forma que ele seja capaz de levantar uma carga de 1kg. Outro ponto de destaque é a futura utilização desse protótipo pelos alunos de engenharia, para o desenvolvimento das habilidades na área de robótica e programação.

No relatório mostrado a seguir será abordado o desenvolvimento do braço robótico, as partes físicas e tecnológicas que o compõe, como as peças mecânicas, servo motores, fonte de alimentação, *LaunchPad* MSP430, aplicativo para Android, o programa *App Inventor* e a comunicação Bluetooth. Será abordado cada um desses itens individualmente, dando uma introdução teórica através de textos, imagens ou diagrama de blocos. Serão mostrados os materiais utilizados, um embasamento do valor gasto para o desenvolvimento do protótipo, as dificuldades encontradas, os pontos fortes e os pontos que ainda precisam de alguma melhoria, os resultados obtidos e a conclusão tirada após o término do projeto.

MATERIAIS E MÉTODOS

Estrutura mecânica:

A estrutura mecânica, cujos desenhos estão ilustrados nos anexos, foi desenvolvida com base em um TCC já desenvolvido na universidade [20] , utilizando de pequenas adaptações de acordo com as particularidades do protótipo.

As dimensões da base fixa e móvel foram as mesmas utilizadas no TCC citado anteriormente. As peças que formam a estrutura do braço, incluindo os eixos, foram projetadas de forma que não fosse utilizado muito material para não tornar o sistema pesado, porém que pudesse suportar uma carga considerável sem que houvesse uma distorção deste material conforme pode ser observado no formato das peças do número 3 a 8 na seção de anexos deste relatório. Devido à complexidade no desenvolvimento da garra, esta foi comprada e acoplada junto ao braço. Para as conexões, foram utilizados parafusos de rosca máquina com cabeça chata com fenda de diâmetro 3mm e comprimento de 15mm em conjunto com porcas e arruelas. Para auxiliar na movimentação dos eixos, foram utilizados rolamentos de 14mm de diâmetro na parte estrutural do lado oposto ao eixo dos servo motores. Os desenhos foram feitos com o auxílio do software AutoCAD 2010 e a usinagem das peças foi feita pelos mecânicos da UNIVAP, em uma chapa de alumínio de 2mm de espessura, fornecida pela própria instituição de ensino.

App Inventor:

A ideia do desenvolvimento de um aplicativo para o controle do braço robótico surgiu após a descoberta do MIT *App Inventor*. O *App Inventor* é uma ferramenta desenvolvida pela

Google que permite a criação de aplicativos para smartphones que rodam o sistema operacional Android, sem que seja necessário conhecimento em programação [3].

O programa foge das linhas de programação normal e possibilita até mesmo usuários comuns lançarem seus aplicativos. Graças ao recurso *drag and drop* (arrastar e soltar), a programação das aplicações acontece de forma simples e intuitiva [3]. Fornecendo desde as opções gráficas (design de plano de fundo, botões, textos, imagens) até opções de utilização de hardwares do próprio dispositivo a ser instalado (acelerômetro, câmera, reconhecimento de voz, GPS, Bluetooth, etc) [4].

O *App Inventor* é dividido em duas partes, o *App Inventor Designer* e o *App Inventor Blocks Editor*.

O *App Inventor Designer* é a tela inicial do projeto, nela você desenha seu aplicativo, escolhendo a posição dos botões e imagens, inserindo fotos, *droplists*, *checkboxes* e outros componentes disponíveis para a construção de um programa [5], A figura 1 ilustra a tela inicial do Aplicativo desenvolvido para controlar o braço robótico, aplicativo esse denominado Brom.

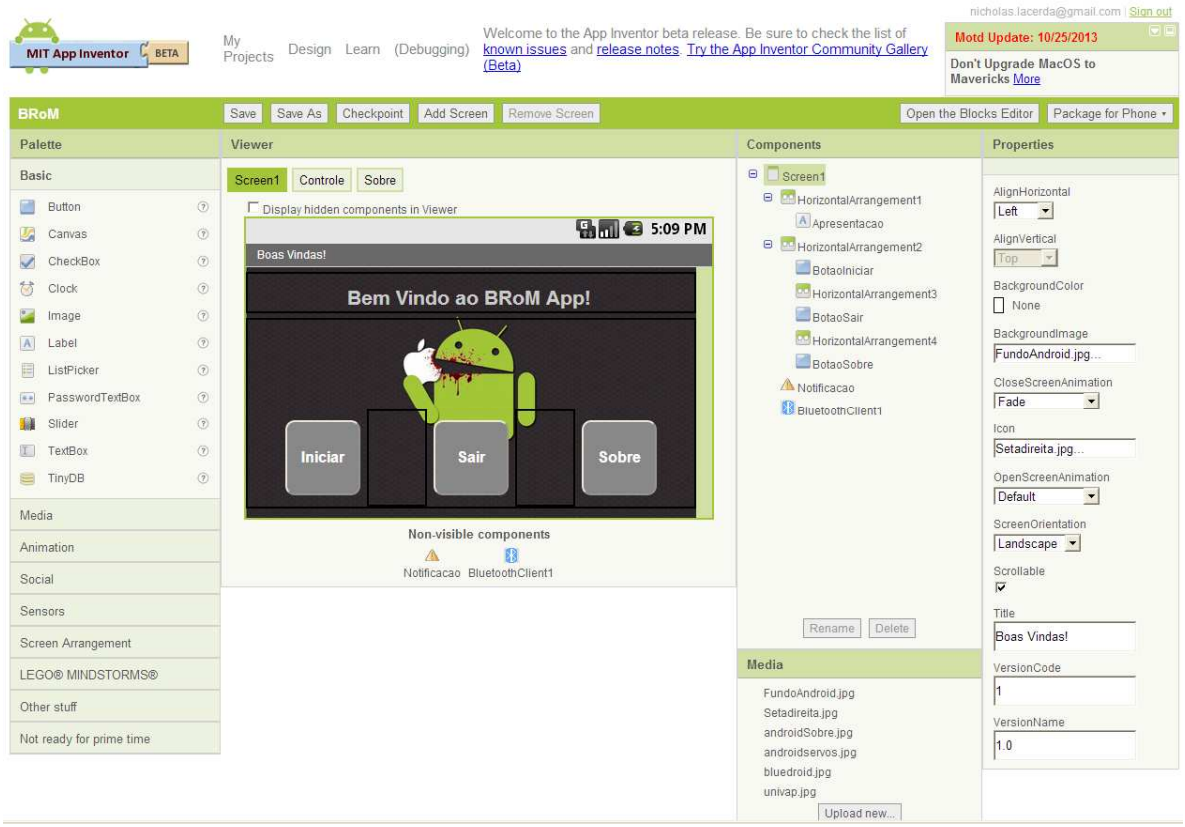


Figura 1 – Tela do Aplicativo Brom, desenvolvido no *App Inventor Designer*

Para utilizar o *App inventor Blocks Editor* é necessário configurar e instalar alguns aplicativos, o primeiro é o Java, em seguida é necessário instalar o aplicativo de desenvolvimento do *App Inventor* [5].

A área de programação dos aplicativos possui uma grande lista de opções para o desenvolvimento dos mesmos onde a linguagem raiz utilizada é o C. Através da montagem de blocos lógicos torna-se o desenvolvimento fácil e de rápida adaptação [4], de acordo com a figura 2.

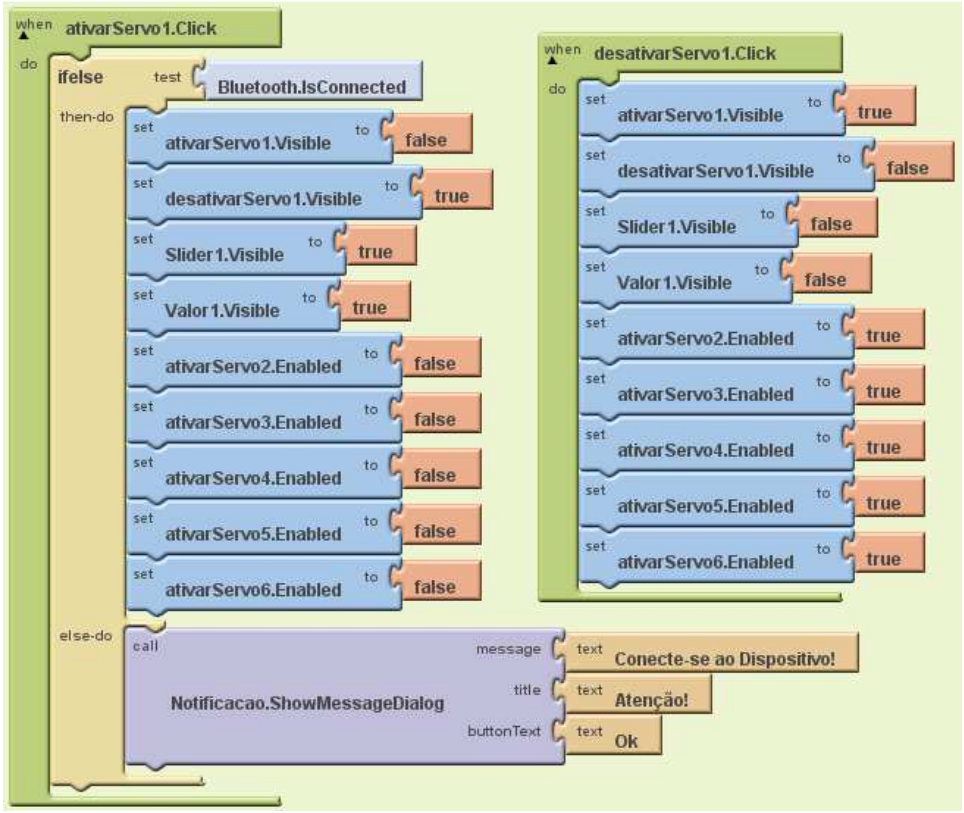


Figura 2 – Programação da tela de controle no *App Inventor Blocks Editor*

Uma vez finalizado, o aplicativo pode ser descarregado diretamente para o dispositivo (*smartphone, tablet*) através de uma conexão USB ou até mesmo via *Wireless* (é necessária a instalação do aplicativo MIT *AICompanion* para realizar a interface via Wi-Fi) [4].

Sabendo das opções que esta ferramenta proporciona, amadureceu-se a ideia de criar um aplicativo que pudesse enviar dados via Bluetooth para um microcontrolador e a partir daí controlar o braço robótico. A figura 3 traz o esquemático da lógica de funcionamento do aplicativo desenvolvido.

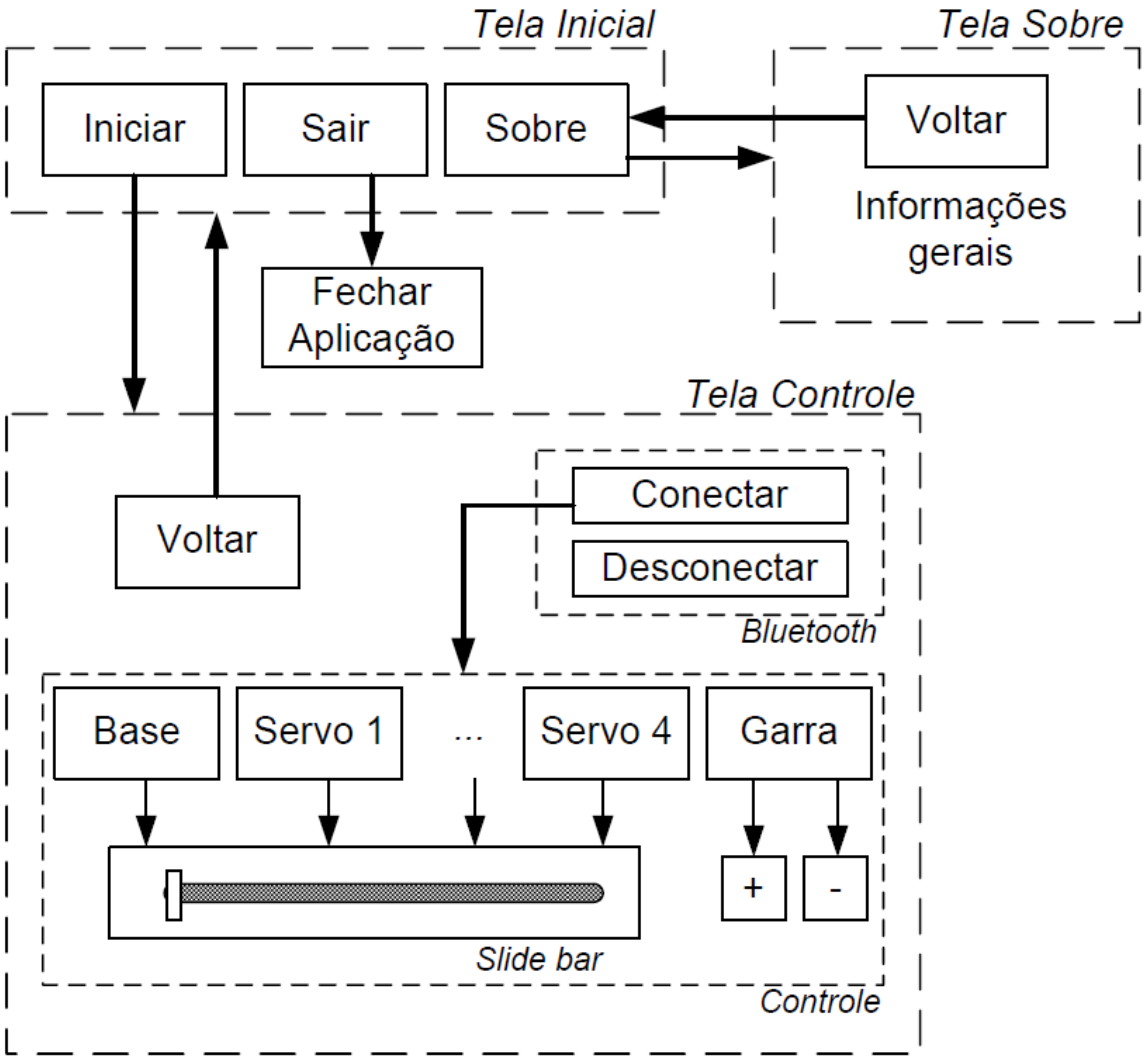


Figura 3 – Esquemático da lógica de funcionamento do aplicativo

Bluetooth:

A comunicação Bluetooth entre o *smartphone* e o MSP430 foi realizada através do módulo JY-MCU, que está ilustrado na figura 4, essa comunicação foi escolhida por se tratar de uma interface de alta confiabilidade na transmissão de dados, de fácil utilização já que a maioria dos smartphones possui esse recurso, por ter um alcance de distância que atende as necessidades do projeto, bastando somente que os dispositivos estejam pareados e permaneçam próximos um do outro, conforme será explicado a seguir.

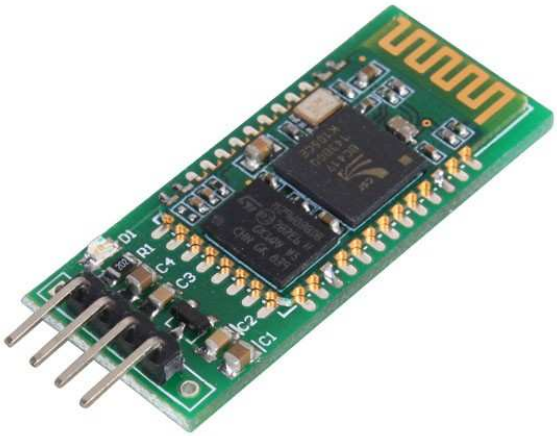


Figura 4 - Módulo Bluetooth

O Bluetooth é uma comunicação sem fio que permite que computadores, smartphones, tablets e afins troquem dados entre si e se conectem a mouses, teclados, fones de ouvido, impressoras e outros acessórios a partir de ondas de rádio. A troca de informações entre dispositivos é feita de maneira rápida, descomplicada e sem uso de cabos, bastando que um esteja próximo um do outro, independente de suas posições [6].

O alcance máximo do Bluetooth é dividido em três classes:

Classe 1: Potência máxima de 100mW, alcance de até 100 metros;

Classe 2: Potência máxima de 2,5mW, alcance de até 10 metros;

Classe 3: Potência máxima de 1mW, alcance de até 1 metro;

Dispositivos de classes diferentes, podem se comunicar normalmente, bastando respeitar o limite daquele que possui o menor alcance [6].

Foi optado pela Classe 2, devido ao alcance atender as necessidades propostas (10 metros) e seu consumo de potência ser relativamente baixo, apenas 2,5mW.

A velocidade de transmissão de dados do Bluetooth é relativamente baixa: até a versão 1.2, a taxa pode alcançar, no máximo, 1Mb/s (megabit por segundo). Na versão 2.0, esse valor passou para até 3Mb/s. Embora essas taxas sejam baixas, são suficientes para uma conexão satisfatória entre a maioria dos dispositivo. Todavia, a busca por velocidades maiores é constante, como prova a versão 3.0, capaz de atingir taxas de até 24MB/s [6].

O Bluetooth é uma tecnologia criada para ser utilizada no mundo inteiro, por isso é utilizada uma frequência de rádio aberta, aceita em praticamente qualquer local do planeta. A faixa ISM (*Industrial, Scientific, Medical*), que opera à frequência de 2,45GHz, é a que mais

se aproxima dessa necessidade. Como a faixa ISM é aberta, é necessário garantir que o sinal do Bluetooth funcione mesmo com as interferências que possam existir. [6].

O padrão de comunicação FH-CDMA (*Frequency Hopping - Code-Division Multiple Access*), utilizado pelo Bluetooth, permite a proteção contra interferências, já que faz com que a frequência seja dividida em vários canais. O dispositivo que estabelece a conexão vai mudando de um canal para outro de maneira muito rápida. Esse esquema é chamado "salto de frequência" (*frequency hopping*). Isso faz com que a largura de banda da frequência seja muito pequena, diminuindo sensivelmente as chances de uma interferência. No Bluetooth, pode-se utilizar até 79 frequências (ou 23, dependendo do país) dentro da faixa ISM, cada uma espaçada da outra por 1 MHz. [6].

Como um dispositivo se comunicando por Bluetooth pode tanto receber quanto transmitir dados (modo *full-duplex*), a transmissão é alternada entre slots para transmitir e *slots* para receber, um esquema denominado FH/TDD (*Frequency Hopping/Time-Division Duplex*). Esses *slots* são canais divididos em períodos de 625 µs (microssegundos). Cada salto de frequência deve ser ocupado por um *slot*, logo, em 1 segundo tem-se 1600 saltos [6].

No que se refere ao enlace, isto é, à ligação entre o emissor e receptor, o Bluetooth faz uso, basicamente, de dois padrões: SCO (*Synchronous Connection-Oriented*) e ACL (*Asynchronous Connection-Less*). O primeiro estabelece um link sincronizado entre o dispositivo master e o dispositivo escravo, onde é feita uma reserva de slots para cada um. Assim, o SCO acaba sendo utilizado principalmente em aplicações de envio contínuo de dados, como voz. Por funcionar dessa forma, o SCO não permite a retransmissão de pacotes de dados perdidos. Quando ocorre perda em uma transmissão de áudio, por exemplo, o dispositivo receptor acaba reproduzindo som com ruído. A taxa de transmissão de dados no modo SCO é de 432 Kb/s, sendo de 64 Kb/s para voz [6].

O padrão ACL, por sua vez, estabelece um link entre um dispositivo mestre e os dispositivos escravos existentes em sua rede. Esse link é assíncrono, já que utiliza os *Slots* previamente livres. Ao contrário do SCO, o ACL permite o reenvio de pacotes de dados perdidos, garantindo a integridade das informações trocadas entre os dispositivos. Assim, acaba sendo útil para aplicações que envolvam transferência de arquivos, por exemplo. A velocidade de transmissão de dados no modo ACL é de até 721 Kb/s [6].

Quando dois ou mais dispositivos se conectam por meio do Bluetooth, eles formam uma rede denominada Piconet. Nessa conexão o dispositivo que iniciou a conexão assume o papel de mestre, enquanto os outros dispositivos assumem o papel de escravo. Cabe ao mestre à tarefa de regular a transmissão de dados na rede e o sincronismo entre os dispositivos [6].

Cada Piconet pode suportar até 8 dispositivos (um mestre e 7 escravos), no entanto, é possível elevar esse número a partir da sobreposição de Piconets. Em poucas palavras, este procedimento consiste em fazer com que uma Piconet se comunique com outra que esteja dentro do limite de alcance, esquema este denominado Scatternet, ambos estão representados na figura 5 [6].

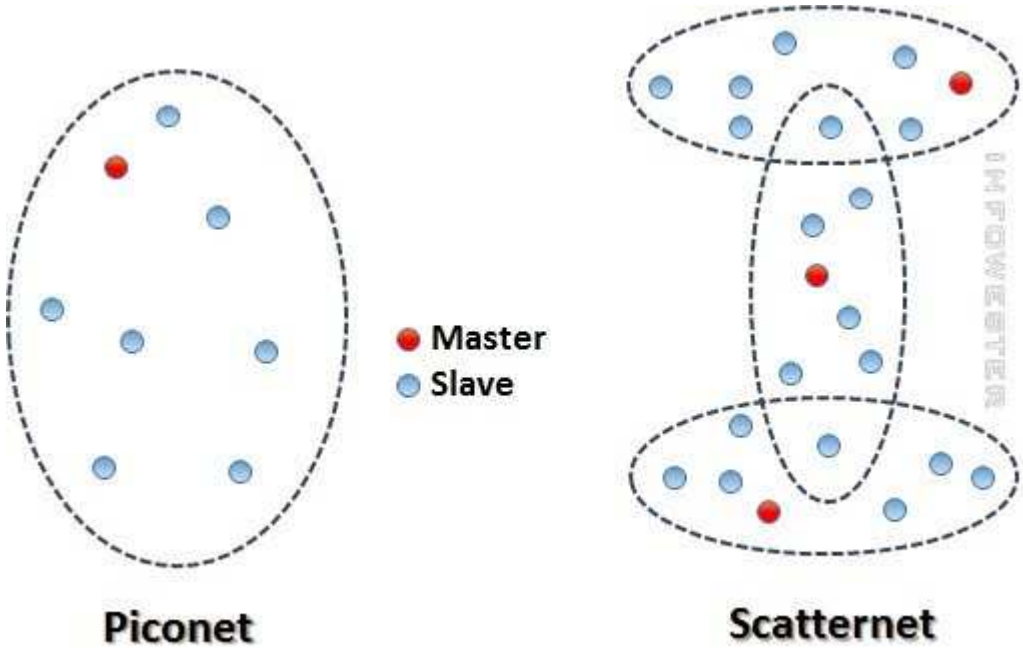


Figura 5 – Rede Bluetooth – Piconet e Scatternet

Para que cada dispositivo saiba quais outros fazem parte de sua Piconet, é necessário fazer uso de um método de identificação. Para tanto, um dispositivo que deseja se conectar a uma Piconet já existente pode emitir um sinal denominado Inquiry. Os dispositivos que recebem o sinal respondem com um pacote FHS (*Frequency Hopping Synchronization*), informando a sua identificação e os dados de sincronização Piconet. Com base nestas informações, o dispositivo pode então emitir um sinal chamado *Page* para estabelecer a conexão com outro dispositivo. Como o Bluetooth é um dispositivo que também oferece economia de energia como vantagem, um terceiro sinal denominado *Scan* é utilizado para fazer com que os dispositivos que estiverem ociosos entrem em *standby*, poupando energia.

No entanto, dispositivos neste estado são obrigados a “acordar” periodicamente para checar se há outros aparelhos tentando estabelecer conexão [6].

Microcontrolador:

A programação do braço robótico foi realizada em linguagem C no Microcontrolador da *Texas Instruments*, o MSP430G2553.

Alguns fatores foram levados em consideração para escolha deste microcontrolador, como por exemplo:

- Baixo consumo: São conhecidos pelo seu consumo incrivelmente baixo (ordem de 0,1µA para retenção de dados na RAM, 0,8µA para funcionamento no modo de relógio de tempo real e cerca de 250µA/MIPS em funcionamento normal). O Baixo consumo é obtido graças aos diversos modos de funcionamento da CPU [8].

- Baixa tensão de operação: Pode operar com tensões a partir de 1,8V até 3,6Volts (a tensão mínima para programação da FLASH é 2,2V [8].

- Alto desempenho: Utilizando um barramento de 16 bits (a maioria dos concorrentes utilizam chips de 8 bits), diversos modos de endereçamento e um conjunto de instruções pequeno, mas muito poderoso, permitem realizar tarefas complexas com um código bastante pequeno e rápido [8].

- Conjunto de instruções ortogonais: A disponibilidade de qualquer modo de endereçamento para qualquer instrução e qualquer operando permite que se escrevam códigos pequenos e eficientes, facilitando a tarefa dos compiladores de linguagens de alto nível como a linguagem C [8].

- Número reduzido de instruções: Apenas 27 instruções físicas (*op-codes*) e mais 24 instruções emuladas (variações das 27 instruções que utilizam os geradores constantes), resultando um conjunto de 51 instruções [8].

- Grande quantidade de periféricos – Contam com um conjunto bastante extenso de periféricos internos, com um ênfase especial para conversores AD de até 16 bits em famílias maiores utilizando o conceito “Sigma Delta”, conversores DA, comparador analógico, amplificador operacional programável, controladores de DMA, *timers* com diversos modos de funcionamento (incluindo PWM), controlador de LCD, USARTs com capacidade de endereçamento, multiplicador por hardware com capacidade de executar operações de multiplicação e acumulo[8].

- Facilidade de gravação e depuração: A utilização da Interface JTAG (do acrônimo inglês *Joint Test Action Group*) para gravação e depuração permite que o projetista realize a programação e a depuração do seu software diretamente na placa de aplicação em tempo real, sem a necessidade de utilização de equipamentos dispendiosos como emuladores [8].

Na tabela 1 estão algumas características importantes desse microcontrolador:

Tabela 1 - Principais características do MSP430 utilizado [7]

Característica	**Dados**
Frequência (MHz)	16
Flash (kb)	16
SRAM (kb)	0,5
GPIO	24
Cap Touch I/O	Sim
Timers – 16 bits	2
Watchdog	Sim
Módulo de reset por queda de tensão)	Sim
USCI_A (UART/LIN/IrDA/SPI)	1
USCI_B (I2C & SPI)	1
Comparadores	Sim
Sensor de temperature	Sim
Conversor A/D	10-bit SAR
Conversor A/D Canais	8
BSL (Bootstrap Loader)	UART

Comunicação Serial:

A comunicação pelo Bluetooth se dá através da interface serial.

Serial é um protocolo muito comum para comunicação de dispositivos que vem como padrão em quase todo PC. A maioria inclui duas portas seriais baseadas em RS-232. Serial também é um protocolo de comunicação muito comum que é utilizado por muitos dispositivos para instrumentação. Além disso, a comunicação serial pode ser utilizada para aquisição de dados em conjunto com um dispositivo remoto de amostragem [10].

O conceito de comunicação serial é bem simples. A porta serial envia e recebe bytes de informação um bit de cada vez. Embora esta seja mais lenta que a comunicação paralela, que

permite a transmissão de um byte inteiro por vez, ela é mais simples e pode ser utilizada em distâncias maiores [10].

Normalmente a serial é utilizada para transmitir dados ASCII. A comunicação serial é complementada utilizando 3 linhas de transmissão: (1) Terra, (2) Transmissão, e (3) Recepção. Visto que a serial é assíncrona, a porta está apta a transmitir dados em uma linha enquanto recebe dados em outra. As características importantes da serial são taxa de transmissão (*Baud Rate*), bits de dados (*datas bits*), bits de parada (*Stop Bits*), e paridade. Para duas portas de comunicação, estes parâmetros devem corresponder: [10]

- Taxa de transmissão (*Baud Rate*): Uma medida de velocidade para comunicação. Isto indica o número de bits transmitidos por segundo. Por exemplo, 300 Baud, são 300 bits transmitidos por segundo. Quando nos referimos a um ciclo de *clock* nós medimos a taxa de transmissão. Por exemplo, se o protocolo pedir uma taxa de transmissão de 4800, então o *clock* está rodando a 4800 Hz. Isto significa que a porta serial está amostrando a linha de dados a 4800Hz. As taxas de transmissão comuns para linhas telefônicas são 14400, 28800 e 33600. Taxas de transmissão maiores que estas são possíveis, mas estas taxas reduzem a distância que cada dispositivo pode estar separado. Estas altas taxas de transmissão são utilizadas para comunicação de dispositivos quando os dispositivos estão próximos, é um típico caso com dispositivos GPID. [10]

- Bits de dados (*Data bits*): Uma medida dos bits de dados atuais e uma transmissão. Quando o computador envia um pacote de informação, a quantidade de dados pode não ser um 8 bits completo. Os valores padrão para pacotes de dados são 5,7, e 8 bits. Qual configuração você deve escolher depende de qual informação você está transferindo. Por exemplo, o standard ASCII possui valores de 0 a 127 (7 bits). O *extended* ASCII usa de 0 a 255 (8 bits). Se os dados forem transferidos em texto simples (*standard ASCII*), então enviar 7 bits de dados por pacote é o suficiente para a comunicação. Um pacote refere-se a uma transferência de byte único , incluindo bits de inicio/fim, bits de dados e paridade. Como o numero atual de bits depende do protocolo selecionado, o termo pacote é utilizado para cobrir todas as instancias [10].

- Bits de parada (*Stop Bits*): Usado para sinalizar o fim da comunicação para um único pacote. Os valores típicos são 1,1.5, e 2 bits. Uma vez que os dados são cronometrados através da linha e cada dispositivo possui seu próprio *clock*, é possível os dois dispositivos virem a estar ligeiramente fora de sincronia. Portanto, os bits de parada não só indicam o fim da transmissão mas também dão aos computadores algumas margem de erro nas velocidades

de *clock*. Quanto mais bits são usados para bits de parada, maior a leniência na sincronização de *clocks* diferentes, mas a taxa de transmissão fica mais lenta [10].

- Paridade: Uma forma simples de verificação de erro que é utilizada na comunicação serial. Há quatro tipos de paridade: par, impar, marcada e espaçada. Naturalmente, a opção de utilizar sem paridade está disponível. Para paridade par e impar, a porta serial irá definir o bit de paridade (último bit depois dos bits de dados) para um valor que garanta que a transmissão tenha um numero par ou impar de bits de lógica alta. Por exemplo, se o dado for 011, então para a paridade par, o bit de paridade será 0 para manter o número de bits de lógica alta par. Se a paridade fosse ímpar, então o bit de paridade será 1, resultando em 3 bits de lógica alta. As paridades marcadas e espaçadas não verificam realmente os bits de dados, mas simplesmente define o bit de paridade alto para paridade marcada ou baixo para paridade espaçada. Isto permite ao dispositivo receptor saber o estado de um bit de modo que habilita o dispositivo determinar se um ruído está corrompendo o dado ou se os *clocks* do transmissor e receptor estão fora de sincronia [10].

Controle PWM:

Para comandar os servos motores foi utilizado o sinal PWM através do microcontrolador.

Os servos motores (será abordado posteriormente) são pequenos motores que possuem internamente circuito de controle, engrenagens redutoras, mecanismo de posicionamento por *feedback* e são extremamente potentes para seu tamanho [9].

Um servo motor possui 3 entradas. Uma de alimentação que é 5 Volts, outra que é GND (0V) e a terceira que é a entrada de controle. Um servo motor é controlado enviando-lhe um sinal PWM (*Pulse Width Modulation*), ou seja, a posição angular irá depender da largura de pulso enviado [9].

O sinal de controle é uma onda quadrada (0 a 5V) de frequência igual a 50Hz (funciona também em 60Hz) [9].

A uma frequência de 50Hz corresponde a um período de 20 ms (1/50=20ms) [9].

O sinal de PWM a enviar terá de ser sempre o primeiro milissegundo a 5V. A partir daí, durante o milissegundo seguinte, o tempo que o sinal se mantiver em 5V irá determinar a posição do servo motor. Ou seja, se imediatamente após o primeiro milissegundo o sinal passar a ser de 0V, então o servo ir-se-á colocar na posição 0º. Se em vez disso, o sinal for durante o primeiro milissegundo igual a 5V e se esse valor se mantiver durante o segundo

milissegundo (igual a 5V) então o servo ir-se-á colocar na posição 180º (a posição angular máxima de servo motor é normalmente de 180º) [9].

Em qualquer dos casos, a partir do segundo milissegundo e durante os 18ms restantes (2ms+18 ms=20 ms) o sinal de controle deve ser de 0V [9].

Assim, pode-se observar que é o tempo durante o segundo milissegundo, que o sinal se mantem a 5V, que determina a posição angular do servo . Ou seja, se o servo tem um raio de ação de 180º então, como 1/180 = 5 µs, bastam incrementos de 5 µs para alterar a posição do servo motor [9].

Exemplos de sinais PWM e respectiva posição angular:

- 1 ms a 5V e 19 ms a 0V corresponde a uma posição angular de 0º;
- 1,005 ms a 5V e 18,995 ms a 0V corresponde a uma posição angular de 1º;
- 1,010 ms a 5V e 18,990 ms a 0V corresponde a uma posição angular de 2º;
- 2 ms a 5V e 18 ms a 0V corresponde a uma posição angular de 180º.

Para controlar o servo motor o sinal deverá ser conforme a figura 6:

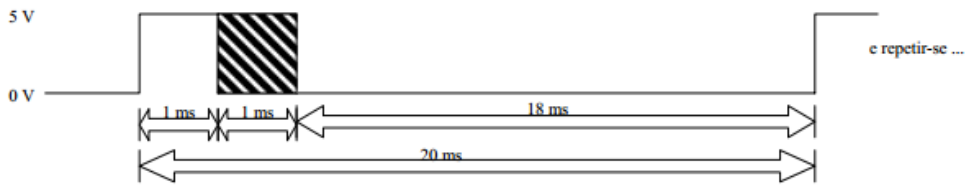


Figura 6 – Sinal para controle PWM do servo motor

Este sinal deve repetir-se para que o servo se posicione na posição angular pretendida e se mantenha lá imóvel, esse dinal é gerado pelo microcontrolador MSP430 [9].

No MSP 430, a geração do sinal PWM é feita através do *Timer* A.

Timer A:

O *Timer* A é um contador/temporizador de 16 bits com as seguintes características [8]:

- Um contador assíncrono progressivo/regressivo de 16 bits com módulo programável e capacidade de interrupção;
- Capacidade de operar a partir de diversas fontes de *clock* interno e externo;

- Até 3 registradores de CCP – Captura/Comparação/PWM, capazes de operar no modo de captura (medição de períodos de sinais), comparação (geração de pulsos de largura programável) e PWM (geração de sinais de frequência e ciclo ativo programáveis);

- Capacidade de captura originada a partir da saída do comparador analógico;

-Possibilidade de iniciar uma conversão A/D a partir da saída de comparação 1 (canal 1).

O princípio de funcionamento do *timer* é relativamente simples. O coração do sistema é o contador de 16 bits (TAR) que é alimentado por um sinal de *clock* selecionável de uma das 4 fontes possíveis TACLK (*Clock* externo), ACLK, SMCLK ou INCLK [8].

A seleção é feita pelos bits TASSELX (registrador TACTL). O sinal pode ainda ser dividido por um fator programável (1,2,4 ou 8), que pode ser selecionado pelos bits IDX (registrador TACTL) [8].

Esse sinal de *clock* provoca o incremento ou decremento da contagem do contador TAR. A direção de contagem é selecionada pelos bits MCx (registrador TACTL). Note que os bits MCx controlam muito mais que apenas a direção de contagem do TAR. Na verdade, eles selecionam o seu modo de operação, conforme mostrado na tabela 2 [8].

Tabela 2 – Modo de operação através selecionado através dos bits MCx

MCX	**Modo**
00	Contagem Parada
01	Contagem de módulo
10	Contagem contínua (de 0 à 0xFFFF)
11	Contagem progressiva / regressiva

No modo 0 (MCx=0), a contagem do TAR permanece parada, mantendo o seu último valor. Nenhum incremento ou decremento acontece nesse modo [8].

No modo 1 (MCx=1), o contador TAR passa a contar progressivamente a cada pulso de clock até atingir o valor programado no registrador TACCR (módulo de contagem). Ao atingir esse valor, o sinal EQU0 é ativado, provocando o reinicio da contagem do TAR. A partir do valor 0. Simultaneamente, o *flag* TAIFG é setado indicando o transbordo da contagem do time [8].

Caso essa interrupção esteja habilitada, o programa será desviado para o devido vetor de interrupção [8].

Observe que caso o programa do usuário altere o registrador TACCR0 para um valor menor que a contagem do TAR, a contagem dela retorna a zero (um pulso adicional de *clock* pode ser necessário antes do efetivo retorno à zero). A figura 7 mostra o diagrama em blocos do *Timer* A [8].

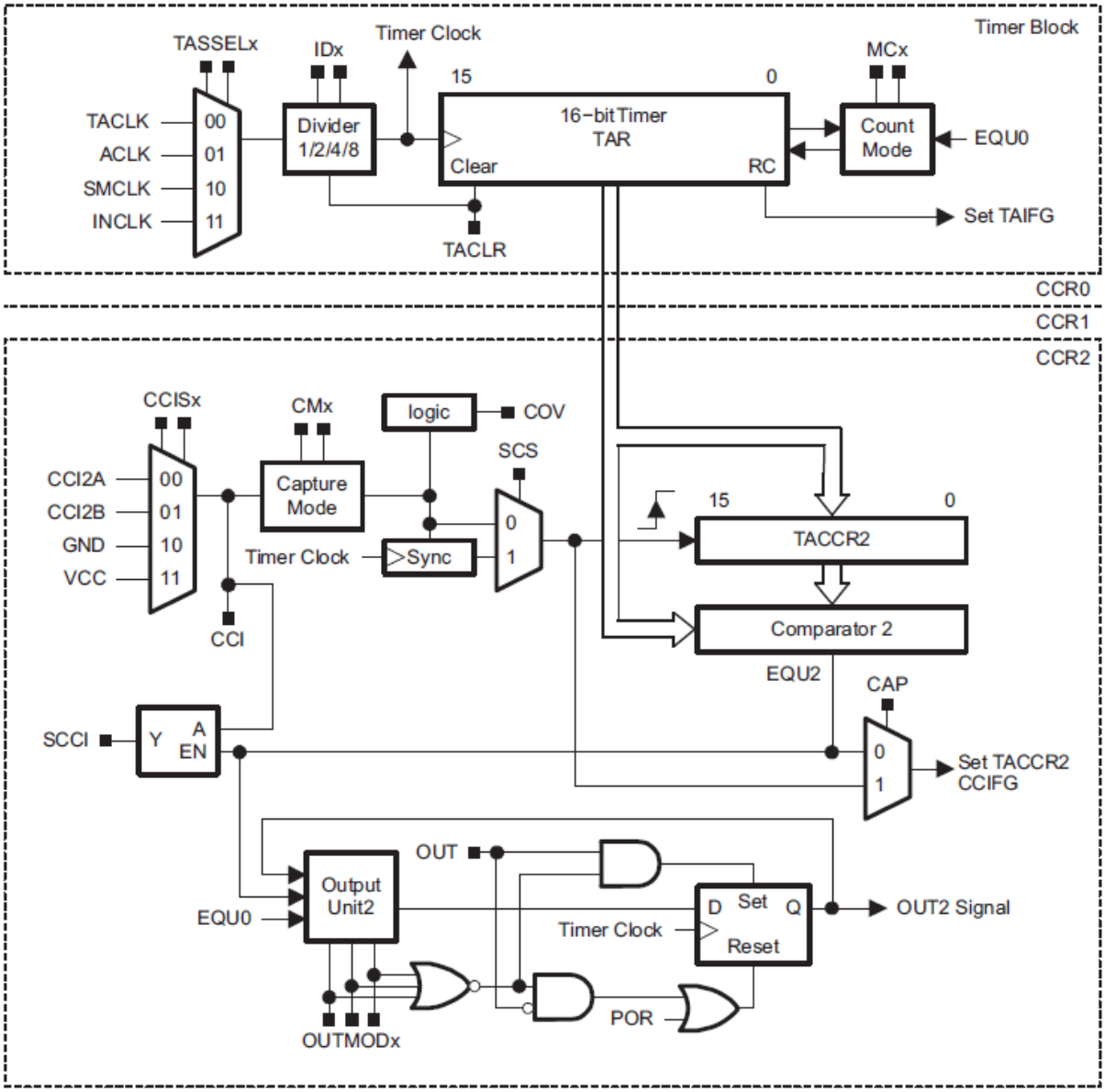


Figura 7 – Diagrama em blocos da estrutura do *Timer* A para três canais de captura/comparação

Servo motor:

Servo motor é uma máquina mecânica ou eletromecânica que apresenta movimento proporcional a um comando, em vez de girar ou se mover livremente. Sem um controle mais

efetivo de posição como a maioria dos motores, servos motores são dispositivos de malha fechada, ou seja: recebem um sinal de controle, verificam a posição atual, atuam no sistema indo para posição desejada. Em contraste com os motores contínuos que giram indefinidamente, o eixo dos servos motores possui uma liberdade de aproximadamente 180º, mas são precisos quanto à posição. Para isso, possuem três componentes básicos: [11]

-Sistema atuador: É constituído por um motor elétrico, embora também possa encontrar servos com motores de corrente alternada, a maioria utiliza motores de corrente contínua. Também está presente um conjunto de engrenagens que forma uma caixa de redução com uma relação bem longa que ajuda a ampliar o torque. O tamanho, torque e velocidade do motor, material das engrenagens, liberdade de giro do eixo e consumo são características chave para especificação de servos motores [11].

Figura 8 – Sistema atuador do servo motor (Motor DC) [12]

Sensor: Normalmente é um potenciômetro acoplado ao eixo do servo. O valor de sua resistência elétrica indica a posição angular em que se encontra o eixo. A qualidade desse sinal vai interferir na precisão, estabilidade e vida útil do servo motor [11].

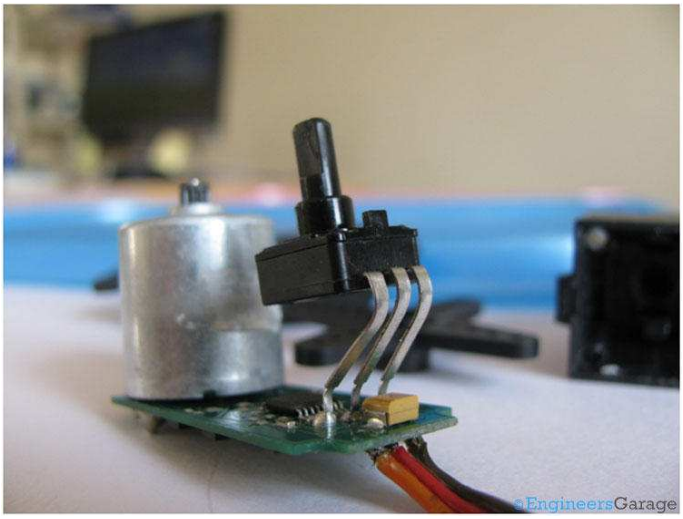


Figura 9 – Sistema de sensoriamento do servo motor [13]

Circuito de controle: É formado por componentes eletrônicos discretos ou circuitos integrados e geralmente é composto por um oscilador e um controle PID (controle proporcional integrativo derivativo) que recebe um sinal do sensor (posição do eixo) e o sinal de controle aciona o motor no sentido necessário para posicionar o eixo na posição desejada [11].

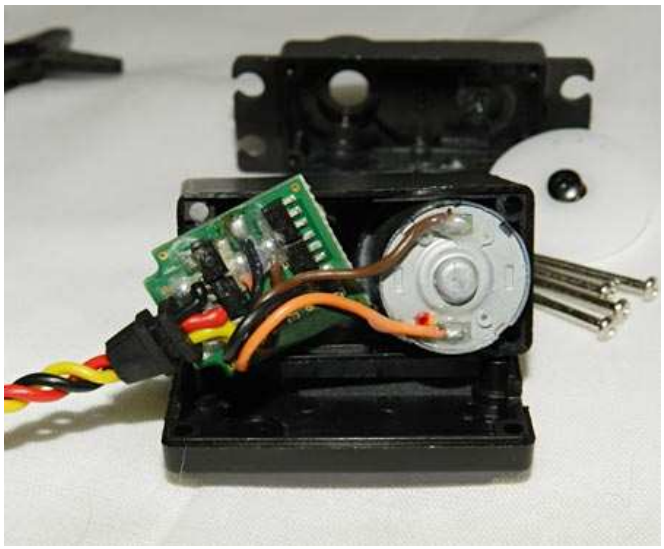


Figura 10 – Circuito de controle do servo motor [13]

Servos possuem três fios de interface, dois para alimentação e um para o sinal de controle. O sinal de controle utiliza o protocolo PPM (Modulação por posição do pulso) que possui três características básicas: Largura mínima, largura máxima e taxa de repetição (frequência) [11].

A largura do pulso de controle determinará a posição do eixo.

Sinal de controle do servo motor: [11]

- Largura máxima equivale ao deslocamento do eixo em +90º da posição central;
- Largura mínima equivale ao deslocamento do eixo em -90º da posição central;
- Demais largura determinam a posição proporcionalmente

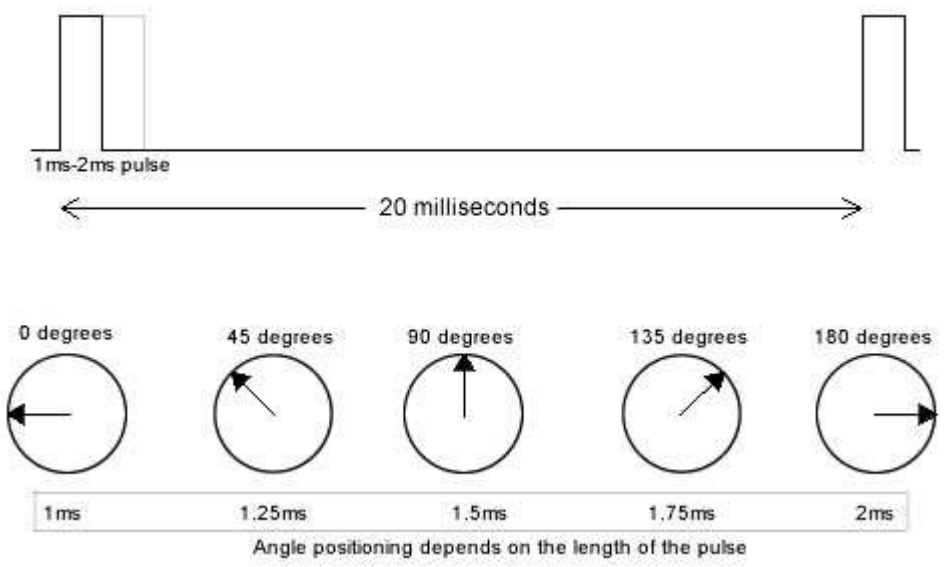


Figura 11 – Sinais de controle do servo motor [14]

Em geral, a taxa de repetição é de 50Hz e a largura do pulso do sinal de controle varia entre 1 e 2 ms, entretanto um servo também pode funcionar a 60Hz [11].

Por existirem diversos tipos de torques oferecidos por seus fabricantes, foi escolhido o emprego destes dispositivos neste projeto uma vez que é exigido precisão, torque e um controle relativamente simples de seus movimentos. Conforme demonstrado a seguir, foi determinado o torque do servo da base do braço uma vez que ele sofre maior esforço no projeto.

Definição do servo motor ideal para o projeto:

Para a definição da especificação dos servos motores utilizados, é necessário saber onde é exigido o torque máximo do braço. Como pode ser observado na figura 12, o braço está na posição onde se exige o maior torque em todo o sistema, que é aplicado no eixo do servo da base móvel (ponto A)

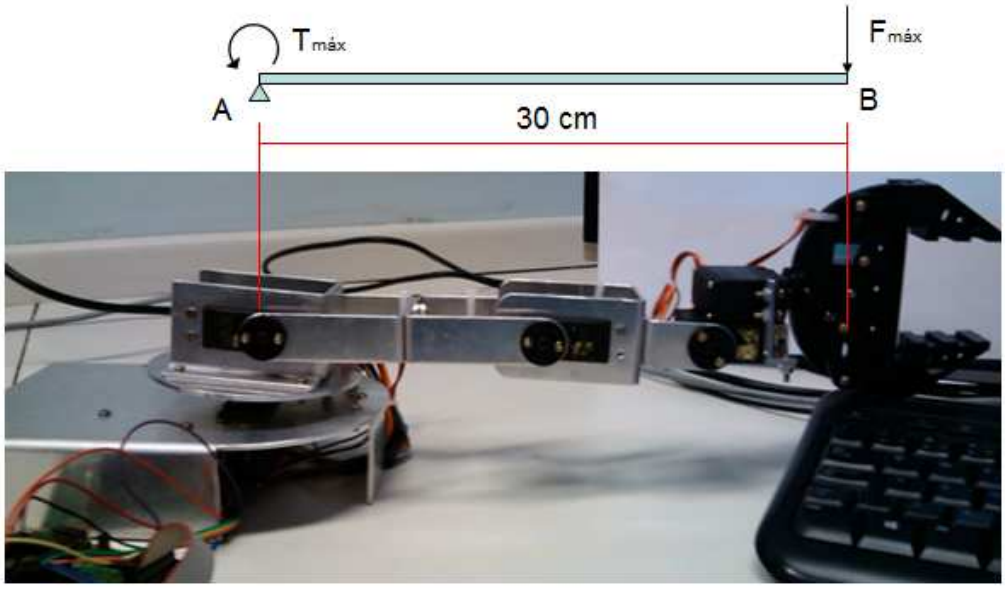


Figura 12 – Especificação dos servo motores

Considerando o pior caso, onde se tem todo o peso de dois servos mais o peso total da garra (incluindo o servo da garra) aplicados na extremidade do braço (ponto B) podemos calcular a $F_{máx}$ e o $T_{máx}$.

Dados:

Peso fornecido pelo fabricante:

- 1 Servo possui 56g [15];
- Garra (incluindo o servo) possui 161g [17];
- Material do braço contém 123g;
- Comprimento total do braço é igual a 30cm.

$F_{máx}$ = (2x56g) + 161g + 123g

$F_{máx}$ = 3801,6N

$T_{máx}$ = 0,030m x 3801,6 = 114,048N.m ou 11,88kg.cm

Logo, através de uma pesquisa de mercado, encontramos o servo motor Tower ProMG995 cujo torque fornecido é de 13kg.cm. Tem-se então um peso máximo de carga que o braço pode levantar que é de 1,2kg.

Uma vez definido o servo com o máximo torque exigido, foi utilizado o mesmo modelo para os demais eixos que demandam um torque menor que o já calculado.

Servo motor MG *Tower Pro* 995:

Uma vez determinado o torque máximo necessário para o sistema, faixa de tensão nominal semelhante ao Launch Pad (para compartilhar a mesma fonte), foi escolhido o servo motor da *Tower Pro*, modelo 995 que une as características requeridas pelo projeto aliadas a um baixo custo sem abrir mão da qualidade de seus componentes como o circuito de controle, potenciômetro e motor DC, estes que são peças chave para o bom desempenho do servo.

Características do MG 995 [15]:

Peso: 56 g

Dimensões: 5.4 cm x 4.4 cm x 2.0 cm

Torque: 13kg/cm

Velocidade de operação: 0.16s/60ºC a 6V

Tensão de operação: 4.8 à 7.2 V

Faixa de temperatura: 0ºC – 55ºC

Ângulo de rotação: 180 graus máximo

Corrente 100 mA

A figura 13 mostra o Servo motor utilizado no braço robótico

Figura 13 – Servo motor *Tower Pro* MG995 [16]

Centralizando o eixo dos servos:

Antes de efetuar a montagem dos servos na estrutura mecânica do braço é necessário encontrar o ponto “zero” de cada motor, ou seja, é preciso que o eixo esteja centralizado no ângulo zero para então ajustar sua montagem em cada junta do braço. Para tal, foi necessário

a utilização de um simples programa descarregado no próprio *LaunchPad* onde este fornece um sinal PWM com uma largura de pulso exatamente entre os pontos máximo e mínimo do servo, desta forma o eixo do servo motor se estabiliza no ângulo zero. Este processo se repete para todos os servos do braço.

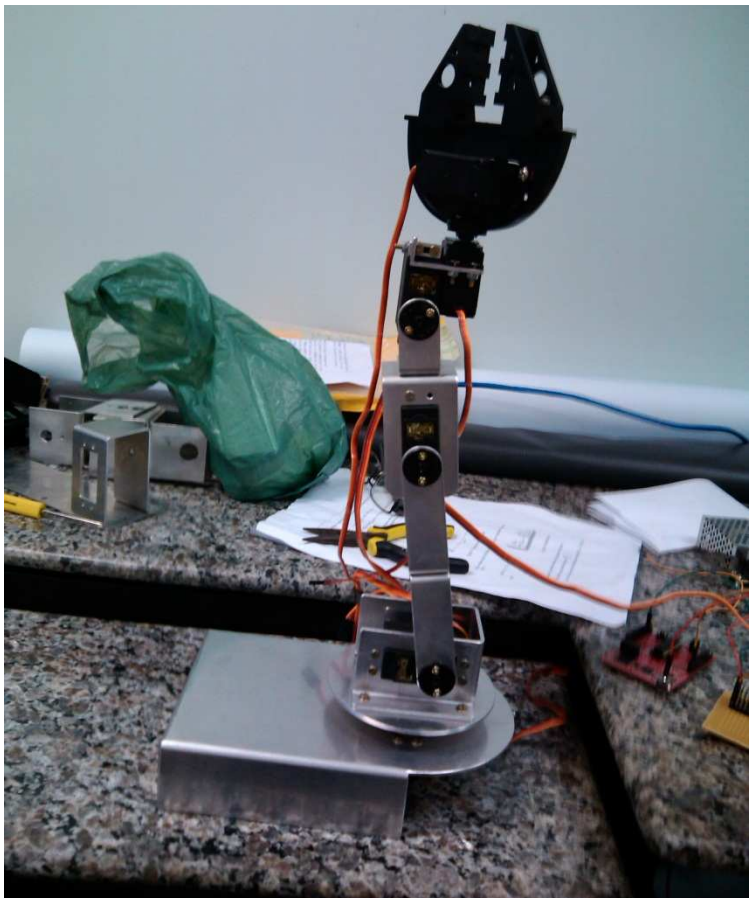


Figura 14 - Servos centralizados

Cálculo da área de trabalho:

Considerando os valores úteis nas medidas do braço, e sabendo que a área de trabalho deste tipo de braço é uma semi-esfera, é possível calcular o volume da região de trabalho. Lembrando que o volume de uma semi-esfera é igual a soma dos volumes dos discos, concêntricos e de espessura infinitesimal, empilhados ao longo do eixo x, de x=r (y=0) até x=0 onde o disco tem raio (y=r), tem-se a seguinte expressão [18]:

$$V\frac{1}{2} = \int_{x=0}^{x=r} \pi(r^2 - x^2)dx$$

Resolvendo a integral tem-se:

$$V\frac{1}{2} = \pi\left(r^3 - \frac{r^3}{3}\right)$$

Logo, o volume de uma semi-esfera é dado pela expressão a seguir [18]:

$$V = \frac{2}{3}\pi r^3$$

A seguir, a figura 15 representa em duas dimensões o arco da semiesfera que representa a área de trabalho do braço:

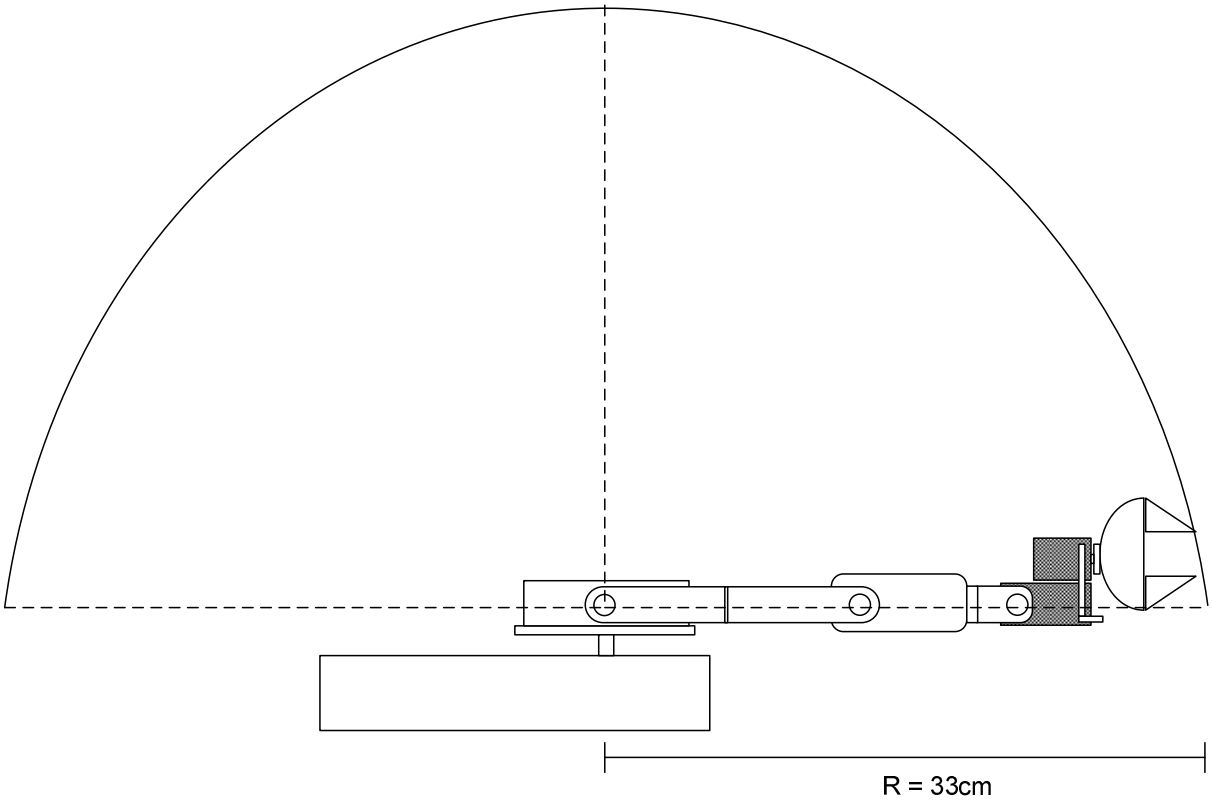


Figura 15 – Área de trabalho do braço robótico

Sabendo que raio r da semiesfera vale 33cm, pode-se calcular o volume da mesma utilizando a equação abaixo [18]:

$$V = \frac{2}{3}\pi r^3$$

$$V = \frac{2}{3}\pi\, 33^3$$

$$\boldsymbol{V = 0,07527\ m^3}$$

Fonte de alimentação:

Os servos motores de torque elevado como os utilizado neste projeto, consomem uma corrente considerável, principalmente nos seus picos ao sair da inércia.

Como o MSP430G2553 não pode fornecer uma corrente necessária para movimentar os servos é preciso que uma fonte de alimentação externa faça esse papel.

Esta fonte é a responsável por alimentar tantos os seis servo motores quanto a *LaunchPad*.

Para determinar os valores nominais da fonte de alimentação é necessário primeiramente saber o consumo dos componentes aos quais serão conectados a ela.

Cada servo motor trabalha na faixa dos 100mA, porém considerando os picos de corrente e acrescentando uma margem de segurança para a fonte, foi determinado que cada servo irá consumir da fonte 450mA.

O MSP430 possui uma grande característica no quesito consumo de energia, devido aos diversos modos de funcionamento de sua CPU, sua corrente de operação é da ordem de µA, no qual pode se tornar desprezível para o dimensionamento da fonte, uma vez que estamos considerando uma margem de segurança em torno dos 150mA para cada servo.

Tem-se então que a corrente mínima para a fonte de alimentação será:

6 Servos x 450mA = 2700mA

Quanto a tensão da fonte, foi levado em consideração a faixa de operação dos componentes, que giram em torno de 4.8 à 7.2V para os servos e 5V para o *LaunchPad* (esta possui um regulador de tensão que ajusta seus valores para o microcontrolador 3.3V).

Após pesquisa de mercado, foi escolhida uma fonte de tensão com as seguintes características:

Tensão de entrada: 110 a 220 VAC (Tolerância de 15%);

Tensão de saída: 5V/3.8A DC

Figura 16 – Fonte de alimentação utilizada no projeto [19]

Fluxo de comunicação do protótipo:

Conforme está demonstrado na figura 17, o *smartphone* com o aplicativo BRom enviará o comando desejado via Bluetooth para o módulo JY-MCU, onde o mesmo transmitirá esse sinal para o *LaunchPad*, que por sua vez executará a lógica programada e enviará os sinais necessários para o posicionamento desejado de cada servo motor.

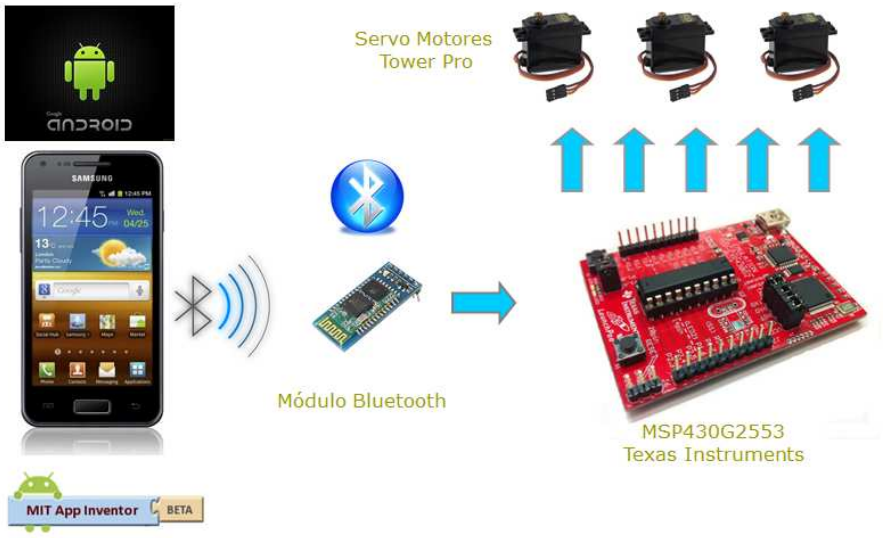


Figura 17 – Fluxo de comunicação do braço robótico

Custo do projeto:

A tabela 3 mostra o custo dos materiais do projeto:

Materiais	**Quantidade**	**Preço Unitário (R$)**
Servo motor Tower Pro. Torque 13kg/cm	4 un	35,18
MSP430G2553	1 un	54,89
Módulo Bluetooth	1 un	24,90
Fonte de alimentação regulada 5V 3.8ª	1 um	18,56
Cabos com conectores	10 m	13,50
Garra articulada equipada com dois servo motores	1 un	97,98
Placa padrão de Fenolite 10x15 cm	1 un	10,00
Total		360,55

RESULTADOS E DISCUSSÃO

Aplicativo Brom:

O Aplicativo foi totalmente desenvolvido e funciona perfeitamente. A conexão Bluetooth e o envio de dados serial com o módulo JY-MCU operam corretamente desde que seja respeitado o limite de distância de 8 metros (limite aceitável da classe 2).

O Brom contém três telas, tela Inicial, tela Sobre e a tela de Controle. A tela Inicial contém três botões:

- Tecla “Sair”: Fecha e encerra o aplicativo;
- Tecla “Sobre”: Abre uma nova tela, apenas com informações adicionais sobre o Brom;
- Tecla “Iniciar”: Clicando sobre ela, será aberta a tela de controle dos eixos e da garra.

A tela de controle, contém 6 botões, sendo 5 botões para controle de cada eixo, que quando clicados abrem uma barra onde o usuário deslizará o dedo sobre a tela até o grau de movimento desejado para aquele eixo (de 0 à 180º), já o 6º denominado “Garra”, quando pressionado, abrirá mais quatro teclas uma com o sinal de “+” e outra com sinal de “-” para que se possa ter um controle mais preciso da abertura e do fechamento da garra, e outras duas “Fechar” e “Abrir” que são responsáveis pelo fechamento e abertura total da garra, respectivamente. A Figura 18 ilustra a tela de controle.

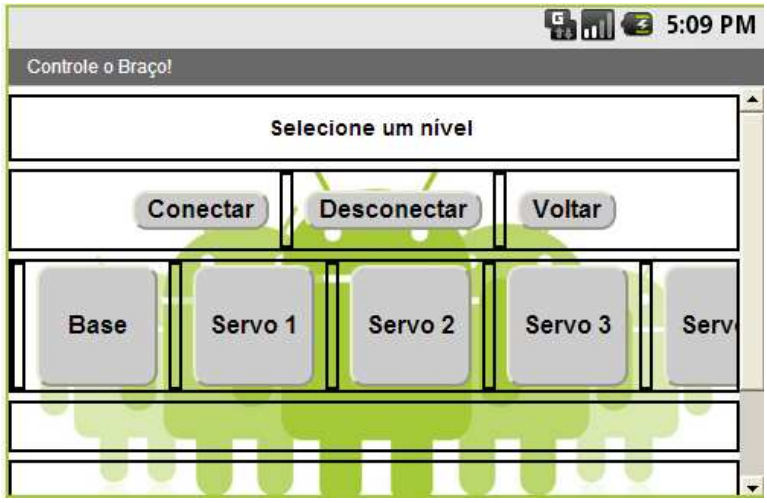


Figura 18 – Tela de controle

Estrutura mecânica:

Ao término da montagem estrutural e ao realizar o teste de movimentação do braço na posição horizontal, como mostra a figura 19, foi observado que a base móvel necessitava de

um apoio para suportar o peso exercido nas bordas da mesma. O problema foi sanado com a inserção de suportes que evitam que todo o peso fique somente no eixo do servo motor, sendo assim, a base não sofre mais com a inclinação excessiva.

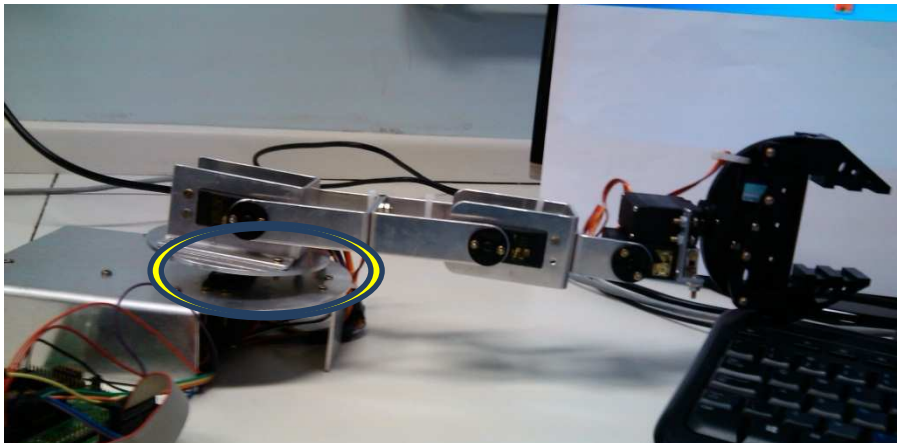


Figura 19 – Inclinação da base móvel

LaunchPad:

Para que fosse possível controlar seis servos motores, foi necessário criar os sinais PWM para os mesmos, utilizando-se de contadores aliados aos timers disponíveis no microcontrolador. A interface serial via Bluetooth funciona corretamente, uma vez que o dispositivo com o aplicativo Brom envia um pacote de dados contendo as informações relacionadas a posição desejada de cada servo motor, a *LaunchPad* por sua vez recebe estes dados e, através do programa, os utiliza para determinar de fato quais os ângulos cada motor irá se deslocar. A programação do MSP430 foi realizada no software IAR *Embedded Workbench* e o fluxograma está na seção de anexos.

Materiais:

Os equipamentos dimensionados neste projeto atenderam as necessidades esperadas ao longo de seu desenvolvimento. A fonte de alimentação suportou a corrente exigida pelos servos sem apresentar falhas ou superaquecimento. Os servos motores foram capazes de gerar um torque excelente sem que houvesse tremulações e sem aquecimento mesmo com uma carga acoplada à garra. A chapa de alumínio de 2mm de espessura utilizada para a estrutura mecânica apresentou bons resultados no que diz respeito ao peso e robustez, já que é um elemento que suporta uma certa carga sem a necessidade de uma grande quantidade de material.

CONCLUSÃO

Este trabalho pode ser divido em três macro sistemas: Aplicativo, Estrutura mecânica e Controlador.

O aplicativo denominado Brom foi desenvolvido a partir de exemplos básicos fornecidos pela própria ferramenta, e após a sua conclusão, pode-se dizer que não há outro aplicativo semelhante, uma vez que o mesmo foi criado visando o controle proporcional especificamente deste braço robótico no qual difere de outros modelos já existentes que funcionam apenas com um movimento pré-definido, tornando o projeto inovador.

A Estrutura mecânica proporcionou, ao longo de seu desenvolvimento, uma grande oportunidade de aprendizado, pois está correlacionado a uma área até então com pouco conhecimento específico, na qual este foi aprimorado durante as pesquisas para que se tornasse possível o desenvolvimento do mesmo. Como exemplo a utilização da ferramenta de design gráfico AutoCAD foi bem aproveitada mesmo com pouco conhecimento do software. Esta estrutura pode ser melhorada em termos de robustez caso seja necessário que ela suporte uma carga maior.

Outro ponto que se tornou uma grande fonte de novos conhecimentos foi a programação do microcontrolador *LaunchPad*, pois as noções de programação na linguagem requerida pelo dispositivo eram limitadas, no qual exigiu um esforço maior para se desenvolver a comunicação entre aplicativo - braço robótico. Pôde-se observar que esta placa de desenvolvimento oferece as mais variadas opções para o programador, dependendo apenas de sua criatividade.

Este projeto oferece diversas oportunidades para continuação de seu desenvolvimento como, por exemplo, a realização de seu controle via Internet no qual o usuário pode controla-lo em qualquer parte do planeta desde que este consiga ter acesso à mesma. Outra aplicação é o acoplamento deste braço com câmeras em uma base móvel onde este pode ser utilizado em trabalhos que coloquem em risco a integridade física de seu operador.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]F. Piltan et al. PUMA-560 Robot Manipulator Position Sliding Mode Control Methods Using MATLAB/SIMULINK and Their Integration into Graduate/ Ungraduate Nonlinear Control, Robotics and MATLAB Courses. International Journal of Robotic and Automation, **Volume 6**. (2012) 106 p.

[2]V. Santos, Robótica Industrial. Aveiro: Universidade de Aveiro, (2004) 1,2- 4,3.

[3]TECH TUDO. **Google App Inventor**. Tech Tudo. 2010. Disponível em <http://www.techtudo.com.br/tudo-sobre/s/google-app-inventor.html> Acesso em 05 Nov.2013

[4]MIT. **Hello Purr**. MIT App Inventor. 2012. Disponível em <http://appinventor.mit.edu/explore/content/hellopurr.html> Acesso em 05 Nov.2013

[5]GUISS, A. **Google APP inventor**: O criador de Apps para Android para quem não sabe programar. Tec Mundo. 2011. Disponível em <http://www.tecmundo.com.br/google/11458-google-app-inventor-o-criador-de-apps-para-android-para-quem-nao-sabe-programar.htm> Acesso em 06 Nov.2013

[6]ALECRIM, E. **Tecnologia Bluetooth**: O que é e como funciona. Info Wester. 2008. Disponível em: <http://www.infowester.com/bluetooth.php> Acesso em 07 Nov. 2013

[7]TEXAS INSTRUMENTS. **MSP 430G2553**. TI. Disponivel em <http://www.ti.com/product/MSP430G2553> Acesso em 08 Nov. 2013

[8]Fabio Pereira. Microcontroladores MSP 430: Teoria e Prática. 1ª.Ed., Editora Érica, São Paulo, (2005) 22, 143,144

[9]MONTEIRO, N ; DINIS, G. **Servo-Motor**. UTAD. 2001. Disponível em: <http://home.utad.pt/~digital2/Apoio/Software/PICS/Servo%20Motor.pdf> Acesso em 11 Nov. 2013

[10]NATIONAL INSTRUMENTS. **Conceitos gerais de comunicação serial**. National Instruments. S.d. Disponível em: <http://digital.ni.com/public.nsf/allkb/32679C566F4B9700862576A20051FE8F> Acesso em 12 Nov. 2013

[11]WIKIPEDIA. **Servomotor**. WIKIPEDIA. 2011. Disponível em: <http://pt.wikipedia.org/wiki/Servomotor> Acesso em 12 Nov. 2013

[12]WISEGEEK. **What is a DC Motor?**. WiseGEEK. S.d. Disponível em: <http://www.wisegeek.org/what-is-a-dc-motor.htm> Acesso em: 12 Nov. 2013

[13]ENGINEERS GARAGE. **How servo motor Works**. Engineers Garage. S.d. Disponível em: <http://www.engineersgarage.com/insight/how-servo-motor-works?page=4> Acesso em 13 Nov. 2013

[14]SINHA, UTKARSH. **How servo motor work?** AI Schack. 2010. Disponível em: <http://www.aishack.in/2010/07/how-servo-motors-work/> Acesso em 13 Nov.2013

[15]DX. **MG995 Tower Pro**. Deal Extreme. 2012. Disponível em: <http://dx.com/p/mg995-tower-pro-copper-servo-gear-for-r-c-car-plane-helicopter-black-173907> Acesso em 14 Nov.2013

[16]TOWER PRO. **MG995 Servo**. Tower Pro. S.d. Disponível em: <http://www.towerpro.com.tw/viewitem1.asp?sn=601&area=50&cat=163#> Acesso em 14Nov.2013

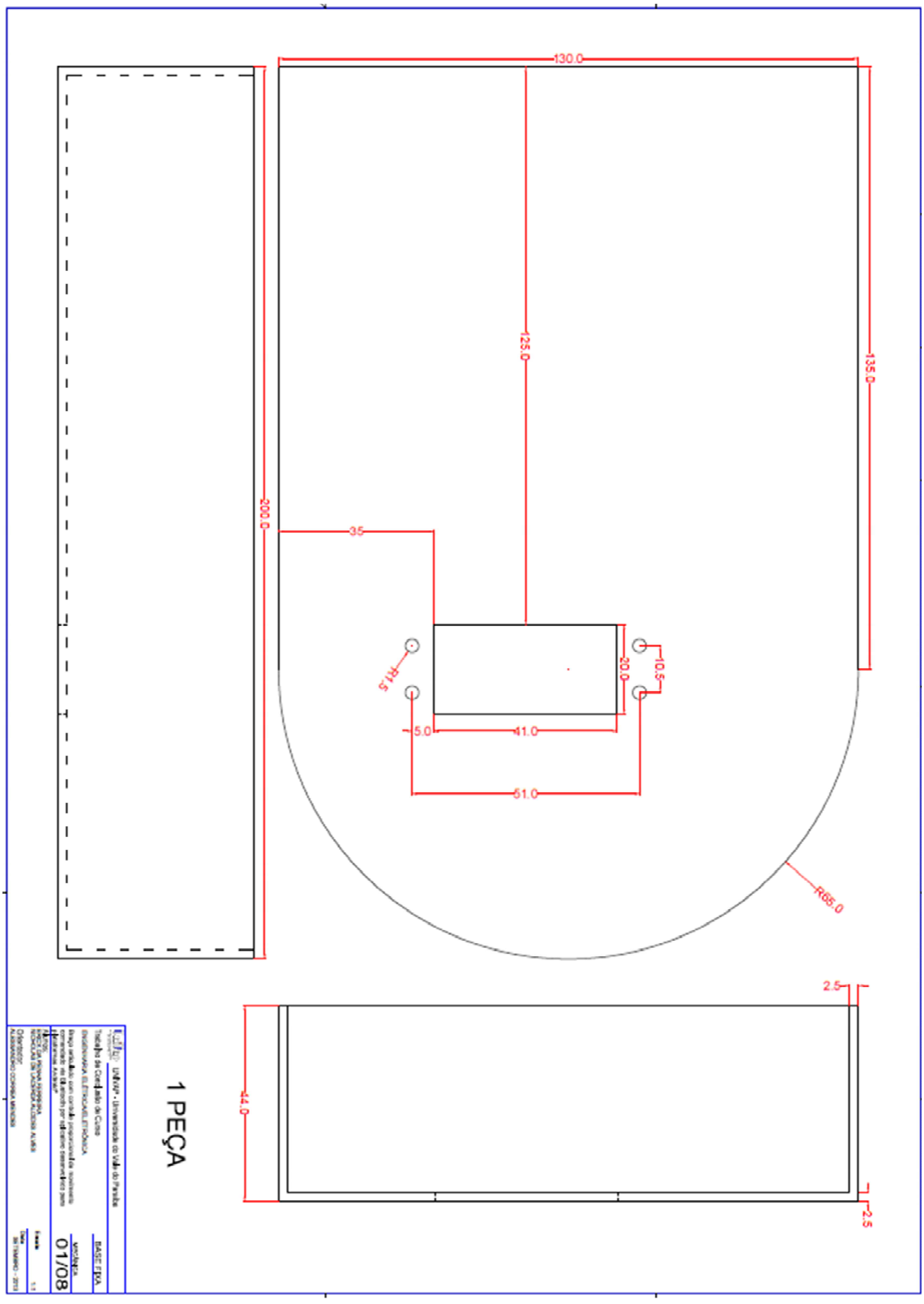
[17]DX. **MG995 Aluminium Alloy Robot Gripper**. Tower Pro. S.d. Disponível em: <http://dx.com/p/mg995-aluminum-alloy-robot-gripper-w-2-servo-for-arduino-works-with-official-arduino-boards-206534> Acesso em 15 Nov.2013

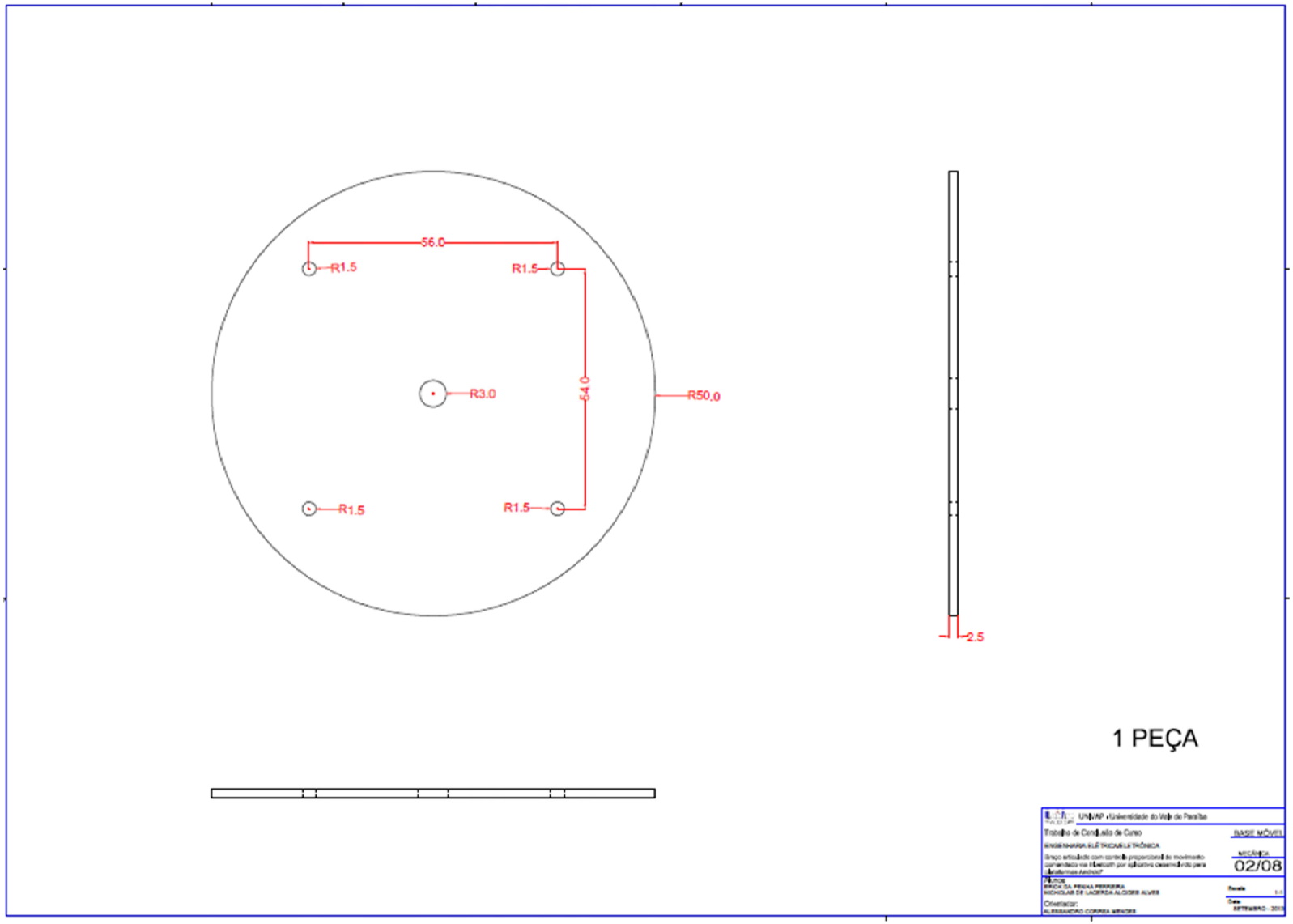
[18]J.I. Venturi, Cônicas e quádricas, 5ªEd.,Editora Unificado, Curitiba, Brasil (2003) 183-185

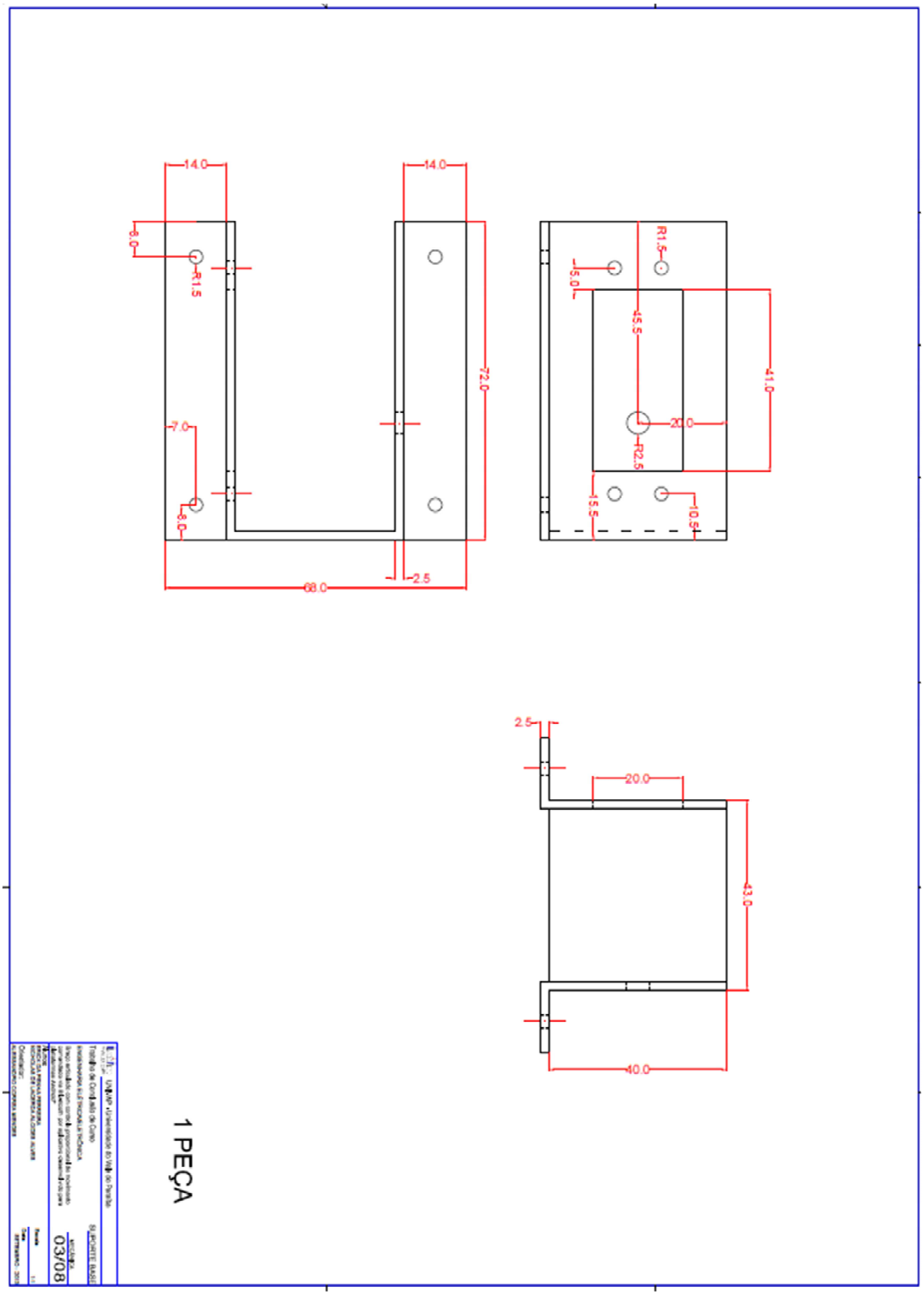
[19]DX. **5V 3.8A Regulated Switching Power Supply**. Tower Pro. S.d. Disponível em: < http://dx.com/p/5v-3-8a-regulated-switching-power-supply-silver-145841> Acesso em 15 Nov.2013

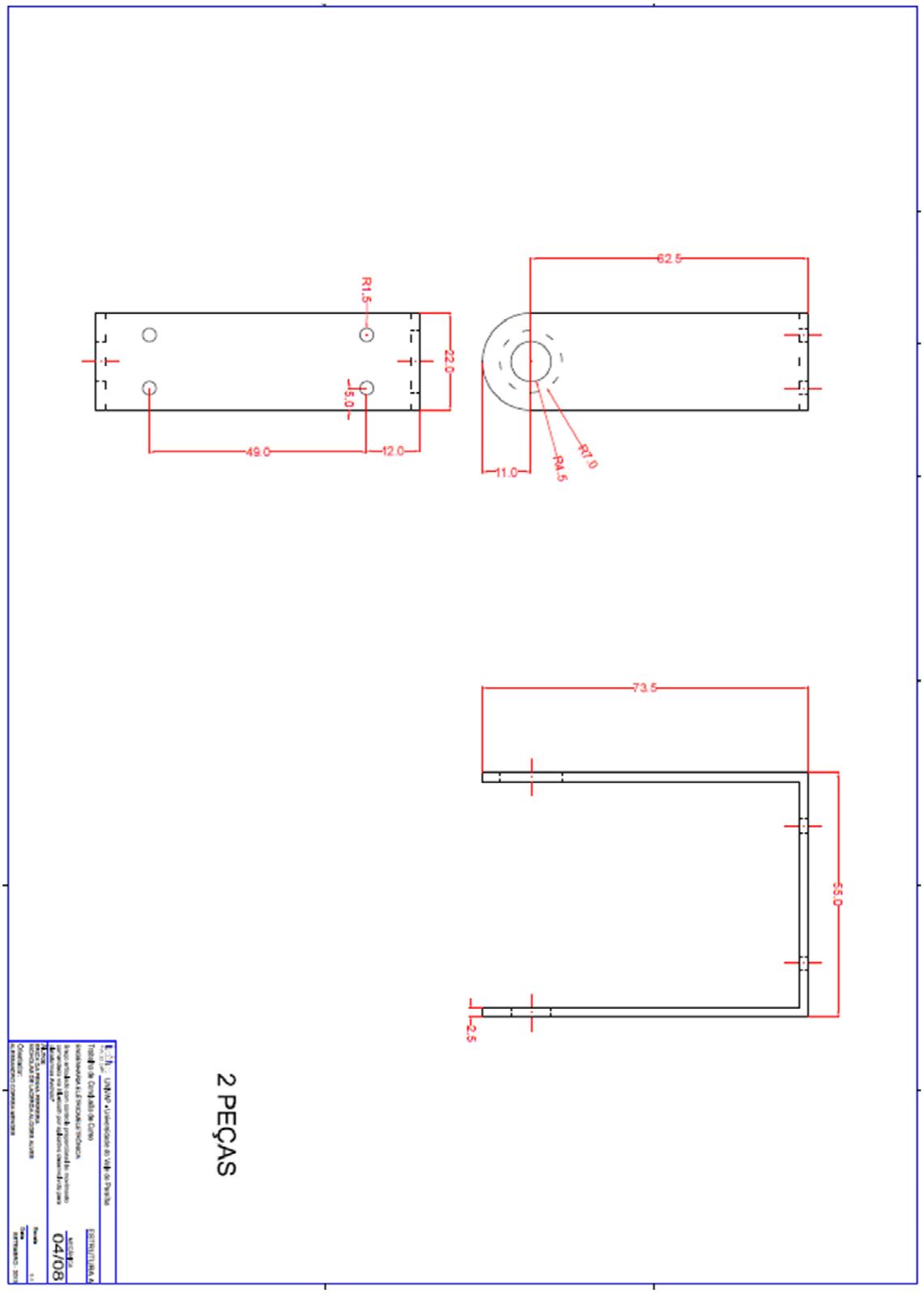
[20]Lopes, G.J.;Oliveira, V.M. Braço articulado controlado remotamente via bluetooth. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica/Eletrônica). São José dos Campos: Faculdade de Engenharias, Arquitetura e Urbanismo, Universidade do Vale do Paraíba, 2013.

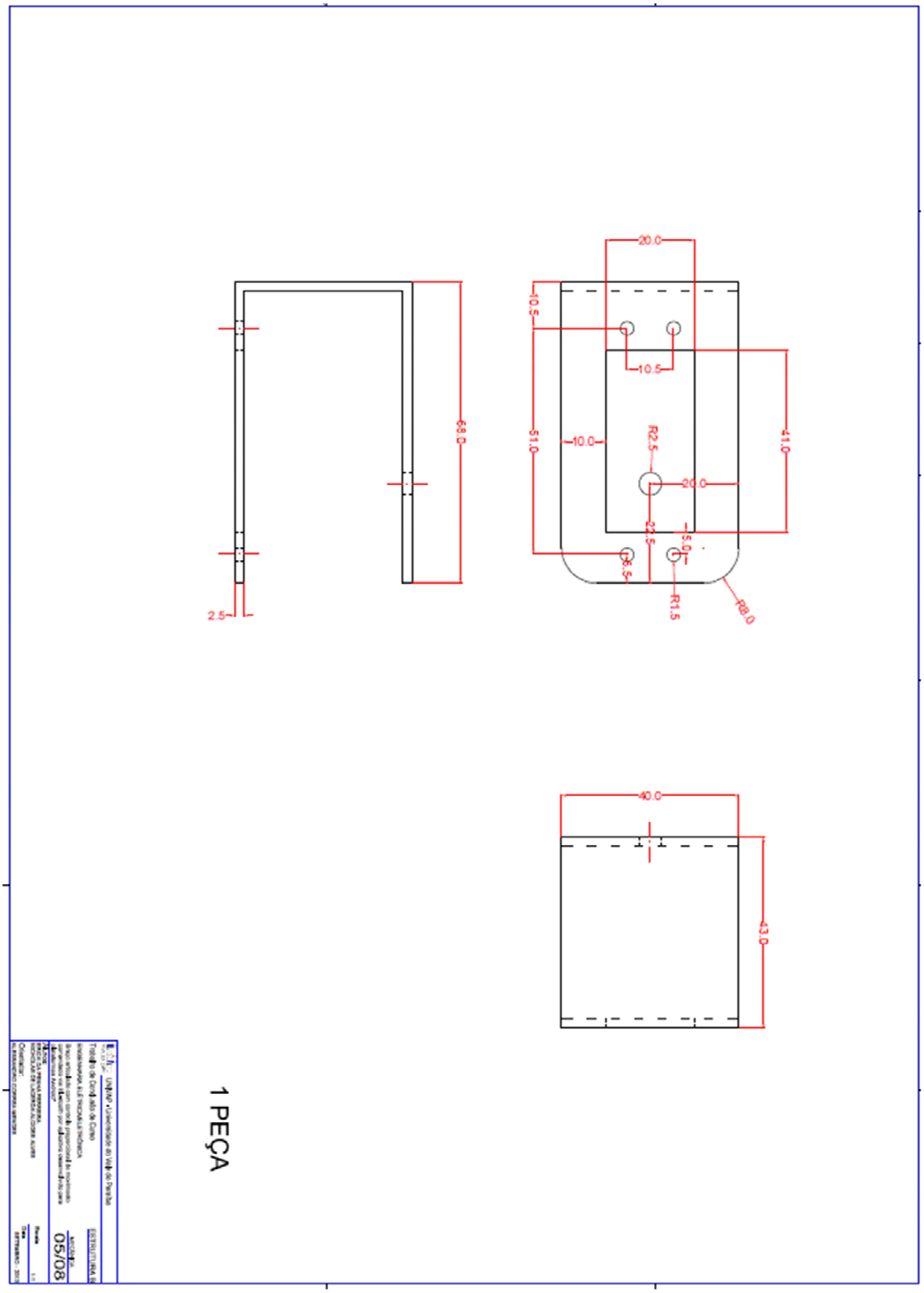
ANEXOS

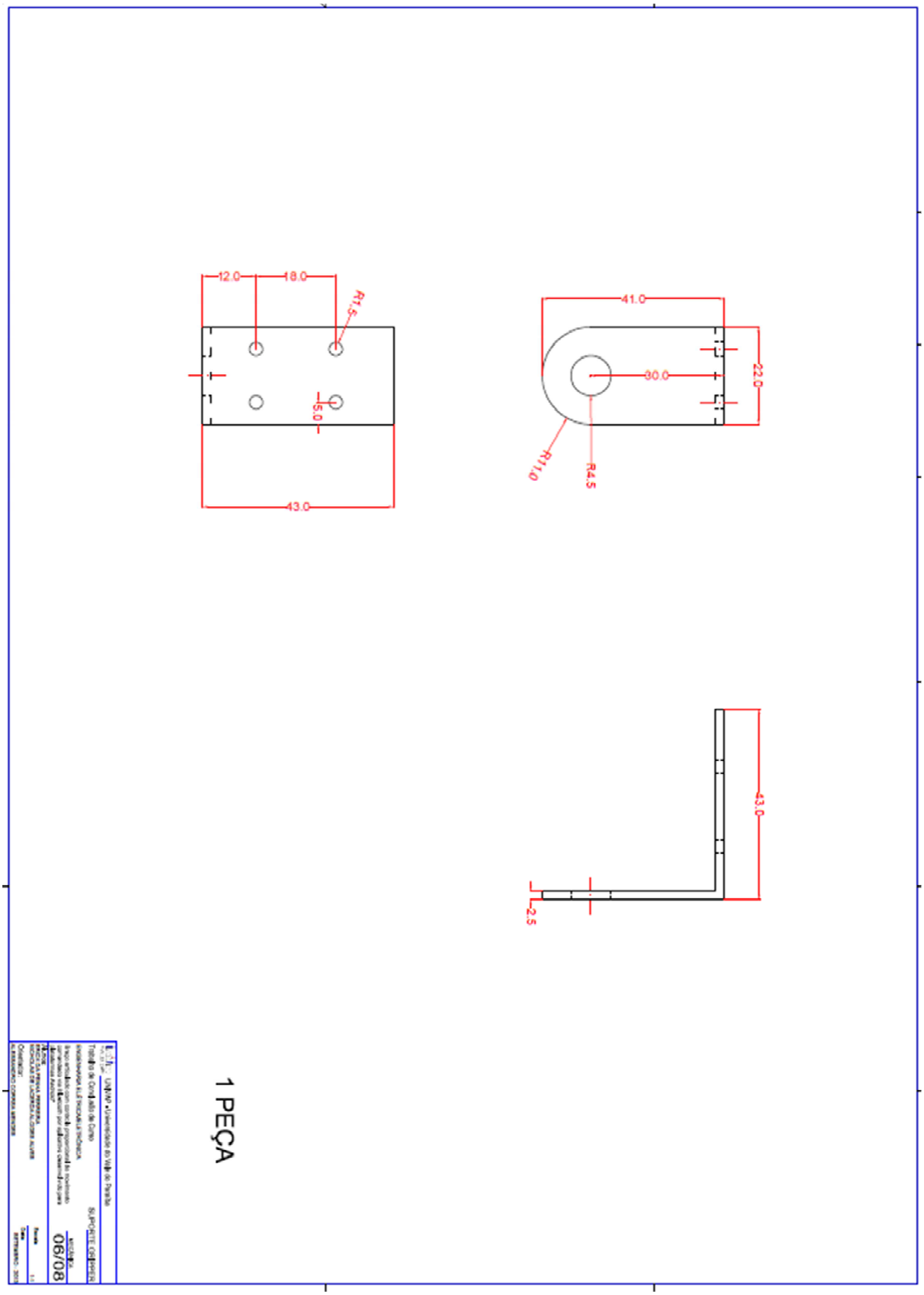

130.0
125.0
135.0
200.0
35
R1.5
20.0
10.5
5.0
41.0
61.0
R65.0
2.5
44.0
2.5
1 PEÇA
BASE EIXA
01/08

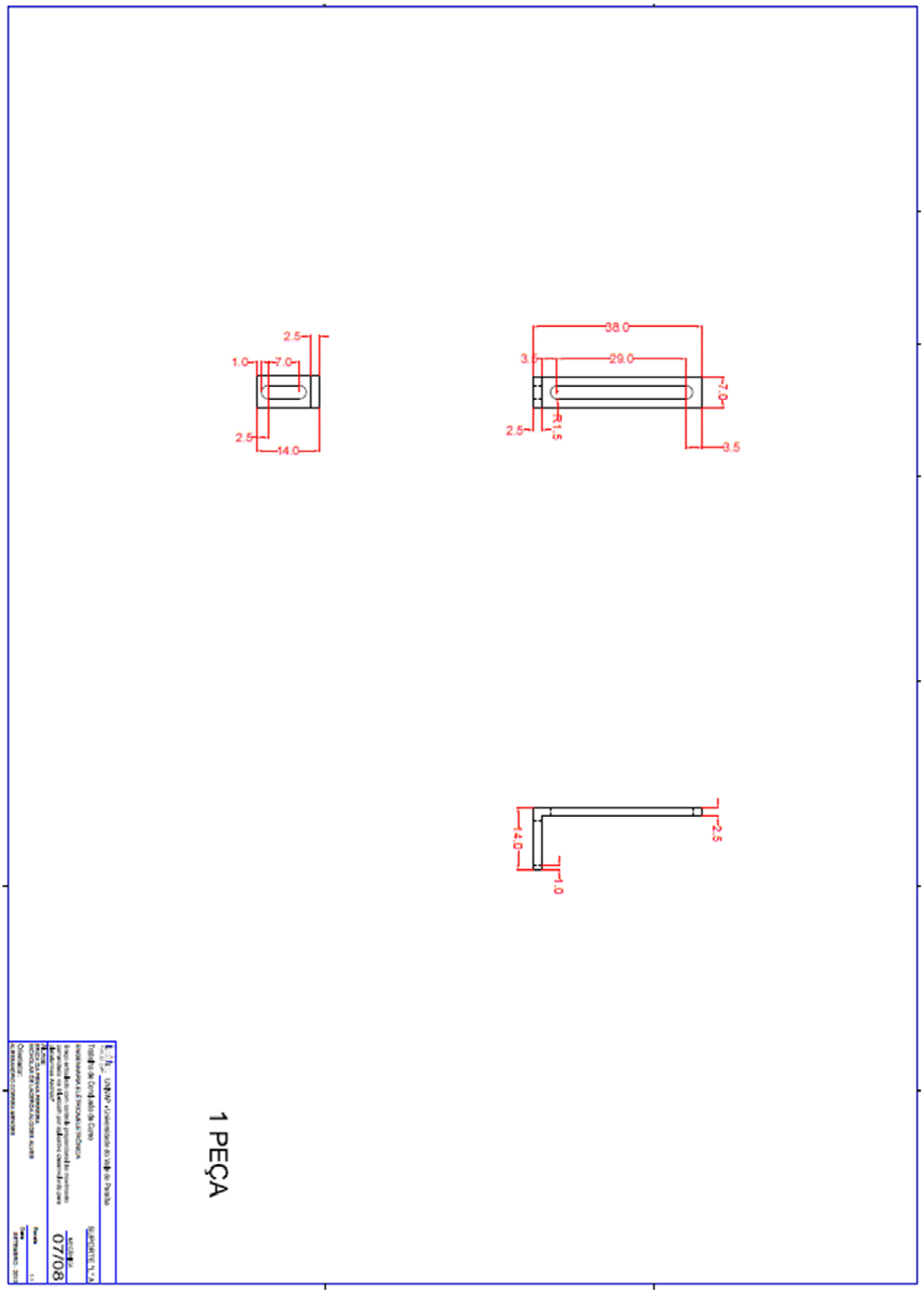

56.0
R1.5
R1.5
R3.0
54.0
R50.0
R1.5
R1.5
2.5
1 PEÇA
UNIVAP - Universidade do Vale do Paraíba
Trabalho de Conclusão de Curso
BASE MÓVEL
ENGENHARIA ELÉTRICA/ELETRÔNICA
MECÂNICA
02/08
Autores:
ERICK DA PENHA PEREIRA
NICHOLAS DE LACERDA ALCOBIA ALVES
Orientador:
ALEXSANDRO CORREA MENDES
Revisão:
Data:
SETEMBRO - 2015

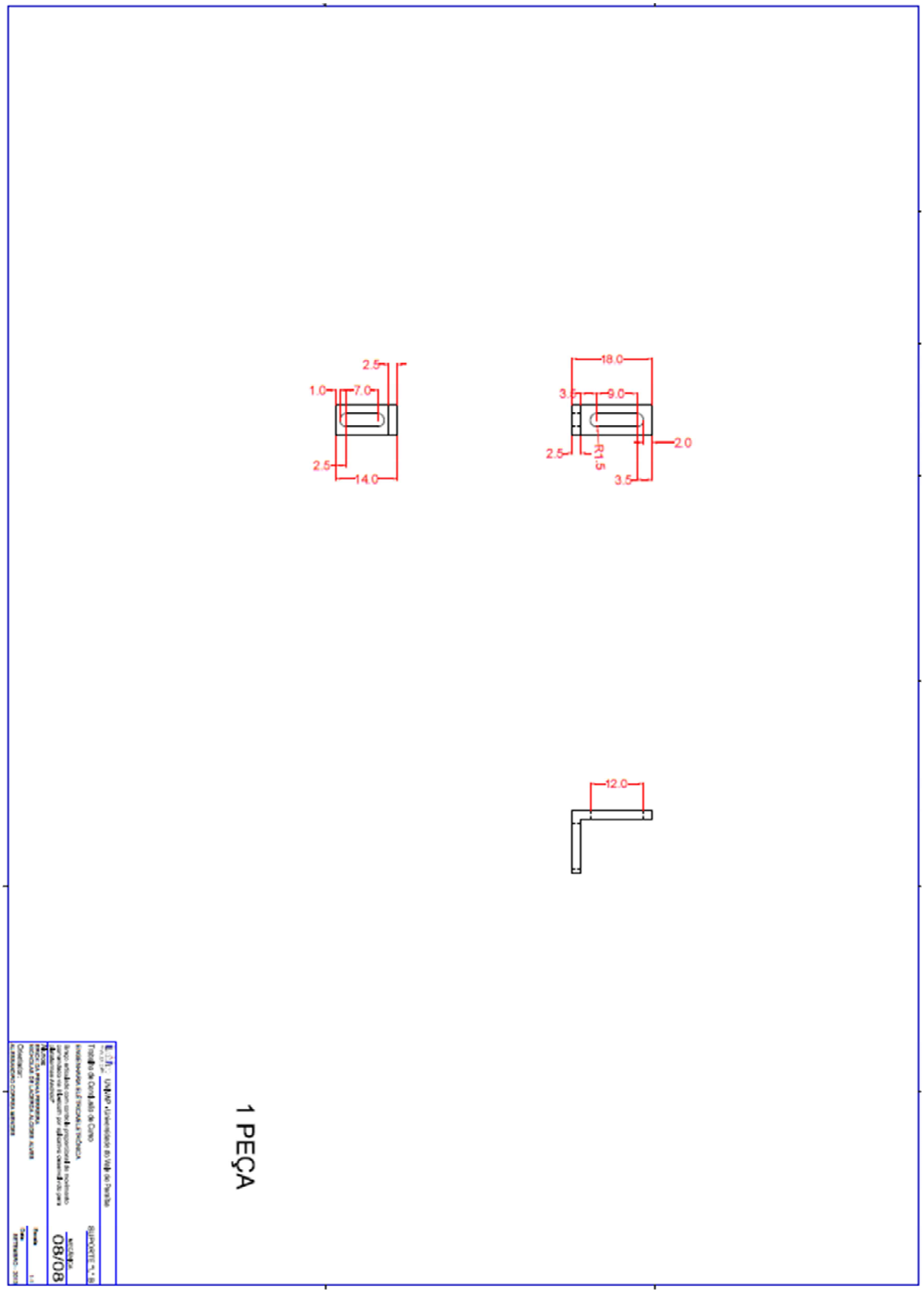

14.0
14.0
6.0
R1.5
72.0
7.0
6.0
68.0
2.5
R1.5
5.0
45.5
41.0
20.0
R2.5
15.5
10.5
2.5
20.0
43.0
40.0
1 PEÇA
03/08


R1.5
22.0
5.0
49.0
12.0
62.5
11.0
R4.5
R1.0
73.5
55.0
2.5
2 PEÇAS
04/08

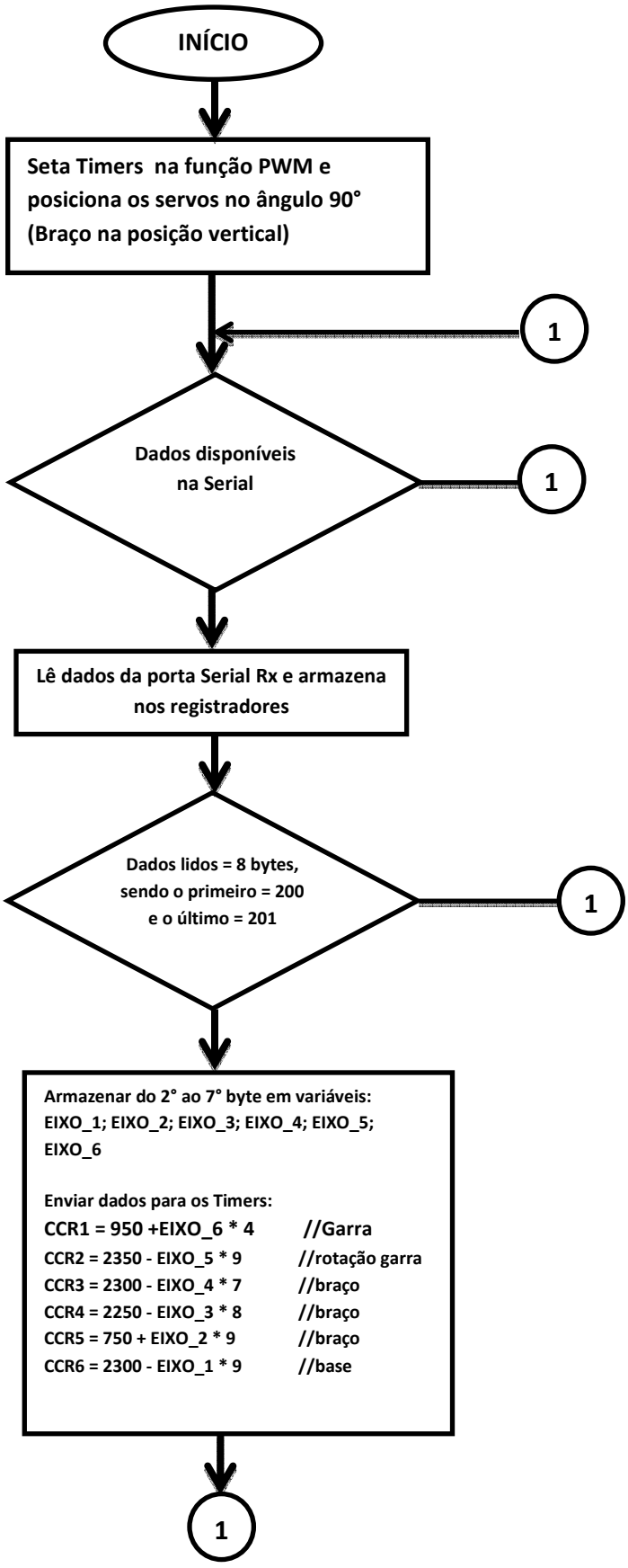

20.0
10.5
10.5
10.0
51.0
R2.5
20.0
41.0
22.5
5.0
6.5
R1.5
R8.0
68.0
2.5
40.0
43.0
1 PEÇA
05/08


12.0
18.0
R1.5
5.0
43.0
41.0
30.0
22.0
R10
R4.5
43.0
2.5
1 PEÇA
UNIVAP - Universidade do Vale do Paraíba
SUPORTE COMPRES
06/08


2.5
1.0
7.0
2.5
14.0
38.0
3.5
29.0
7.0
2.5
R1.5
3.5
14.0
2.5
1.0
1 PEÇA
07/08


2.5
1.0
7.0
2.5
14.0
18.0
3.5
9.0
2.5
R1.5
2.0
3.5
12.0
1 PEÇA
08/08

FLUXOGRAMA


INÍCIO
Seta Timers na função PWM e posiciona os servos no ângulo 90° (Braço na posição vertical)
1
Dados disponíveis na Serial
1
Lê dados da porta Serial Rx e armazena nos registradores
Dados lidos = 8 bytes, sendo o primeiro = 200 e o último = 201
1
Armazenar do 2° ao 7° byte em variáveis: EIXO_1; EIXO_2; EIXO_3; EIXO_4; EIXO_5; EIXO_6
Enviar dados para os Timers:
CCR1 = 950 +EIXO_6 * 4 //Garra
CCR2 = 2350 - EIXO_5 * 9 //rotação garra
CCR3 = 2300 - EIXO_4 * 7 //braço
CCR4 = 2250 - EIXO_3 * 8 //braço
CCR5 = 750 + EIXO_2 * 9 //braço
CCR6 = 2300 - EIXO_1 * 9 //base
1