UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA ELÉTRICA

GIULIA ÂNDREA FERRI

DESENVOLVIMENTO DE UM KIT DIDÁTICO MICROCONTROLADO PARA ENSINO DE ROBÓTICA COM BASE NO ROBÔ FANUC LR MATE 200IC

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2018

GIULIA ÂNDREA FERRI

DESENVOLVIMENTO DE UM KIT DIDÁTICO MICROCONTROLADO PARA ENSINO DE ROBÓTICA COM BASE NO ROBÔ FANUC LR MATE 200IC

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina Trabalho de Conclusão do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Luís Fernando Caparroz Duarte

CORNÉLIO PROCÓPIO
2018

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica

FOLHA DE APROVAÇÃO

Giulia Andrea Ferri

Desenvolvimento de um kit didático microcontrolado para ensino de robótica com base no robô FANUC LR MATE 200IC

Trabalho de conclusão de curso apresentado às 13:00hs do dia 27/09/2018 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____

Prof(a). Dr(a). Luis Fernando Caparroz Duarte - Presidente (Orientador)

_____

Prof(a). Me(a). Ângelo Feracin Neto - (Membro)

_____

Prof(a). Dr(a). Marcio Aurelio Furtado Montezuma - (Membro)

_____

Prof(a). Dr(a). Flávio José de Oliveira Morais - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

Dedico este trabalho à Deus, aos meus pais Denilson e Simone, à minha irmã Giovanna e a todos os meus amigos.

AGRADECIMENTOS

Agradeço primeiramente à Deus, sem Ele nada seria possível.

Ao meu orientador Prof. Dr. Luís Fernando Caparroz Duarte, pelo conhecimento compartilhado e a paciência durante a execução deste projeto.

Aos meus pais, irmã e toda a minha família, pelo apoio imensurável ao longo da minha graduação.

Aos meus amigos, por tornarem tudo mais fácil e alegre.

E a todos que, direta ou indiretamente, contribuíram para tornar este trabalho possível.

RESUMO

FERRI, Giulia Ândrea. **Desenvolvimento de um kit didático microcontrolado para ensino de robótica com base no Fanuc LR Mate 200iC.** 2018. 85 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

O objetivo deste trabalho foi desenvolver um kit para ensino de robótica, que permita ao usuário ganhar experiência na utilização do robô, assim como aprender as leis que regem seus movimentos; utilizando materiais de baixo custo disponíveis no mercado nacional. Para isso foi projetado um dispositivo microcontrolado que funciona como um *Teach Pendant* e que se comunica com o *software* 3DAutomate da *Visual Components*.

O dispositivo projetado permite que as pessoas sejam instruídas sem a necessidade de usar o verdadeiro sistema robótico, estando então em um ambiente muito mais seguro. Pode ser empregado em treinamentos industriais e ser implementado para auxiliar no progresso acadêmico também em cursos online. Neste trabalho o robô Fanuc LR Mate 200iC foi selecionado para um estudo de caso.

Foram estudadas as teorias cinemáticas da robótica para entender os comandos do *Teach Pendant*, e desta maneira, desenvolver um projeto que respeite o modo de operação do robô e todas as rotinas de segurança necessárias.

Palavras-chave: Robótica. Didática. *Teach Pendant.* Microcontrolador.

ABSTRACT

FERRI, Giulia Ândrea. **Development of a microcontrolled teaching kit for teaching robotics based on the Fanuc LR Mate 200iC.** 2018. 85 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

The objective of this project was to develop a robotic teaching kit that allows the user to gain experience in robot use, as well as to learn the laws that govern their movements; using low-cost materials available in the domestic market. To do so, a microcontrolled device that works as a Teach Pendant and communicates with Visual Components software 3DAutomate was designed.

The gadget allows people to be instructed in a much safer environment, without the need to use the true robotic system. It can be used in industrial training and be implemented to aid in academic progress, also in online courses. In this work the Fanuc LR Mate 200iC robot was selected for a case study.

The kinematic robotic theories were studied to understand the teach pendant commands, and in this way, to develop a project that respects the robot's operating mode and all necessary safety routines.

Keywords: Robotic. Teaching. Teach Pendant. Microcontroller.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com o alto desenvolvimento tecnológico recente, o campo da robótica está cada vez mais popularizado e difundido. O que antes só existia nos países mais desenvolvidos, hoje é visto com frequência nos países emergentes, como o Brasil. Conforme SANTOS (2018), com o intuito de aumentar a produtividade e qualidade de seus produtos, as indústrias estão buscando cada dia mais utilizar robôs em seus sistemas produtivos.

Enquanto as empresas buscam métodos para se tornar mais tecnológicas, os trabalhadores se esforçam para aprender a manusear as novas ferramentas e garantir sua sobrevivência no mercado de trabalho. De acordo com MORENO e FAJARDO (2013) um crescimento de 81% foi observado pela busca por educação superior no Brasil.

Porém mesmo na faculdade, muitos não chegam a ter a oportunidade de utilizar recursos parecidos com os encontrados na indústria. O custo de um sistema robótico, por exemplo, é alto e nem todas as instituições possuem laboratórios com robôs disponíveis para seus alunos utilizarem. Segundo DJURIC (2017), mesmo quando existente, a tecnologia é dividida entre todos os alunos e pesquisadores, ou seja, muitos não chegam a utilizar o robô. Surge então um problema: como garantir que estas pessoas estão de fato capacitadas a utilizar a ferramenta, se elas nunca tiveram contato com esta?

Uma solução parcial para o problema, é utilizar *softwares* de simulação que mostram o robô de maneira bem próxima a real e associar com o uso de outras ferramentas. Com este intuito, foi desenvolvido em conjunto com a *Wayne State University* e a *Visual Components* duas versões virtuais de *Teach Pendant*, a interface responsável pela comunicação entre o usuário e o robô*:* uma na tela do computador quando o 3DAutomate é utilizado e a outra um aplicativo para celulares.

Porém, estas ferramentas ainda não possibilitam a experiência do aluno de ter um *Teach Pendant* em mãos e acesso a todas os protocolos de segurança necessárias para fazer o robô se movimentar, tão importante para conhecer o risco associado ao manuseio. Outra alternativa então, é utilizar o *software* como uma ferramenta conectada à um kit didático que torne a experiência mais próxima da realidade.

A utilização de kits didático baratos e de fácil acesso, pode ser a chave para a educação superior de qualidade para muitos. Além da possibilidade de serem empregados em cursos técnicos, tecnólogos e inclusive, cursos a distância. Segundo BALMANT (2018) a quantidade de polos de ensino superior a distância cresceu 133% neste ano. Estes novos polos vão encarar em breve a necessidade de passar para seus alunos a chance de praticar suas habilidades através de tarefas práticas.

1.1 Delimitação do Tema

Neste trabalho foi projetado um kit didático capaz de se conectar com um *software* robótico e auxiliar os estudantes em seu aprendizado. Para isso o kit funciona como um *Teach Pendant*, possuindo o mesmo *layout* de teclado, exigindo iguais rotinas de segurança e envia ao computador qual comando foi pressionado. O *software* recebe este comando e movimenta o robô conforme a ordem do operador.

1.2 Problemas e Premissas

Conforme DENSO ROBOT (2018), utilizar um *Teach Pendant* para fazer que um braço robótico realize um movimento específico, ou ainda um conjunto de movimentos, não é uma tarefa fácil. É necessário habilidade e experiência para não cometer erros que podem danificar o robô ou mesmo ferir uma pessoa próxima. Assim, a utilização do kit didático entra como um importante recurso educacional, visto que permite o aprendizado da ferramenta sem a necessidade de se expor à condições inseguras.

Além disso, conforme DJURIC (2017) muitas faculdades e instituições profissionalizantes não possuem sistemas robóticos em seus laboratórios ou o acesso é muito restrito, impedindo seus alunos de obter conhecimento prático tão importante para a formação destes.

As premissas estabelecidas para desenvolver este projeto, foi que o kit poderá ser utilizado como um artificio de ensino em salas de aula, com o intuito de facilitar o entendimento das leis robóticas e o movimento dos robôs; e o custo do protótipo deve ser baixo, para facilitar a réplica por outros grupos de pesquisa.

1.3 Objetivos

O objetivo deste trabalho é desenvolver um kit didático para ensino de robótica utilizando o microcontrolador PIC18F4520 da Microchip e o *software* 3DAutomate como apoio.

1.3.1 Objetivos Específicos

Os objetivos específicos para este projeto são:

- Escolher um microcontrolador adequado para o projeto
- Estudar o microcontrolador escolhido
- Compreender a teoria robótica
- Estudar o robô LR Mate 200iC e seu *Teach Pendant*
- Desenvolver uma ferramenta capaz de se conectar via Comunicação Serial com o 3DAutomate *software*
- Manter a aparência e as funcionalidade o quão próximo possível do *Teach Pendant.*

2 REVISAO BIBLIOGRAFICA

Primeiro, alguns termos devem ser definidos para facilitar o entendimento do projeto e do processo de execução.

2.1 Robôs

A norma internacional ISO 10218-1:2011 (2018), responsável por definir os procedimentos e métodos necessário para garantir a interação com robôs industriais de forma segura, afirma que robô “é uma máquina manipuladora, com vários graus de liberdade, controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial”. E de acordo com ROSÁRIO (2010), é constituído por um braço mecânico motorizado e um computador que controla seus movimentos. Este computador possui em sua memória programas que detalham o curso que o braço deve realizar ou traça o movimento baseado nos comandos recebidos. O braço recebe sinais enviados pelo computador, ativando os motores e possibilitando o movimento. Além disso, para robôs que não possuem uma programação fixa, mas sim podem receber comandos a qualquer momento, é necessária uma interface entre o operador e robô. Essa interface recebe o nome de *Teach Pendant*.

Conforme CRAIG (2012), o braço mecânico consiste numa série de corpos rígidos (links) interligados por juntas que permitem um movimento relativo entre esses corpos, assemelhando-se assim sua forma geral à de um braço humano e às vezes, quase com as mesmas possibilidades de movimento. Estas juntas podem ser de rotação ou prismáticas (de translação). Uma característica importante dos robôs é o grau de liberdade que estes possuem, ou seja, o número de movimentos individuais das articulações. Cada junta define um ou dois graus de liberdade e, assim, o número de graus de liberdade do robô é igual à somatória dos graus de liberdade de suas juntas.

Para definir o curso que o braço deve realizar os computadores utilizam da teoria cinemática. De acordo com CABRAL (2018), a cinemática de um robô manipulador é o estudo da posição, da orientação e da velocidade (linear e angular) do seu efetuador e de suas juntas, também conhecidas por articulações. Existem 2 tipos de cinemática: direta e inversa. Na cinemática direta têm-se a posição das

articulações e deseja-se obter a posição e velocidade do efetuador; enquanto que na cinemática inversa acontece o oposto, a posição e velocidade do atuador são previamente conhecidas e o intuito é definir a posição e velocidade das articulações.

Segundo PINHEIRO, TRINDADE e PANTOJA (2018), existem diversas formas de se calcular a cinemática de um manipulador. As soluções mais utilizadas são os métodos analíticos e numéricos iterativos. A escolha de um método depende de sua aplicação, do tipo de junta e da estrutura utilizada em relação da quantidade de graus de liberdade. Quando se tem acesso ao um processamento computacional adequado, o método numérico é preferível. Por este motivo, na maioria dos controladores modernos de robôs manipuladores, é o utilizado o método numérico de Denavit-Hartenberg.

Conforme CABRAL (2018) explica, a notação de Denavit-Hartenberg baseia-se no fato de que para determinar a posição relativa de duas retas no espaço, são necessários somente dois parâmetros. Onde o primeiro parâmetro é a menor distância medida ao longo da normal comum entre as duas retas e o segundo, é o ângulo de rotação em torno da normal comum que uma das retas deve girar, de forma que fique paralela à outra. Se para definir a posição relativa de duas retas no espaço são necessários dois parâmetros, para definir a posição relativa de dois sistemas de coordenadas (cada sistema representa uma junta, por exemplo), são necessários quatro parâmetros. Isto porque cada sistema de coordenadas é definido por três retas, sendo que conhecendo dois eixos do sistema, o terceiro está automaticamente definido pelas condições de ortogonalidade.

2.2 *Softwares R*obóticos

Existem no mercado inúmeros *softwares* capazes de simular ambientes contendo robôs manipuladores, o que difere estes sistemas é a linguagem de programação e o grau de complexidade que permitem. Entre as linguagens mais utilizadas, podemos citar: C, C++, Python, Perl, Java, LabVIEW, URBI, MATLAB e BASIC.

Estes *softwares* são utilizados na simulação de plantas virtuais completas, contendo o maquinário, as ferramentas e o controle com o intuito de impactar positivamente a eficiência geral do projeto. Pode ser empregado também para análise de viabilidade antes da implementação de uma nova planta ou da alteração do

maquinário, como por exemplo a decisão de ter parte da produção operada por robôs manipuladores. Além disso, há outros inúmeros benefícios da simulação como: custos reduzidos na produção robótica, diagnóstico de problemas, simular diferentes alternativas sem custos extras, robôs e seus componentes podem ser testados previamente, o projeto pode ser simulado por partes, promovendo uma entrega final com maior rapidez.

Por possuir todos estes recursos, muitos destes *softwares* já possuem um sistema preparado para implementação de novas ferramentas, inclusive um kit didático. Desta forma, torna-se necessário verificar se o sistema já possui um modelo do robô pretendido. Em caso afirmativo, realizar as alterações no projeto do equipamento para garantir a comunicação com o kit.

2.3 Microcontroladores

Segundo SOUZA (2017), um microcontrolador pode ser descrito como sendo um pequeno computador composto por um único circuito integrado. Este circuito contém um núcleo de processador, memória e periféricos programáveis de entrada e saída. Microcontroladores podem ser usados em produtos e dispositivos automatizados como os sistemas de controle de automóvel, dispositivos médicos implantáveis, controles remotos, máquinas de escritório, eletrodomésticos, ferramentas elétricas, brinquedos e outros sistemas embarcados. Ao reduzir o tamanho e o custo em comparação a um projeto que usa um dispositivo microprocessado (circuito integrado que realiza as funções de cálculo e tomada de decisão de um computador), microcontroladores tornam-se econômicos para controlar digitalmente dispositivos e processos. De acordo com a MICROCHIP (2017, A) o seu consumo de energia é relativamente baixo, normalmente na casa dos miliwatts e possui habilidade para entrar em modo de espera (modo '*Idle*' ou '*Sleep*' por exemplo) aguardando por uma interrupção ou evento externo, como o acionamento de uma tecla, ou um sinal que chega via uma interface de dados.

2.4 Comunicação Serial

Conforme PAIOTTI (2017) explica, a comunicação serial foi criada na década de 60 por um comitê chamado de *Eletronic Industries Association* (EIA), com o intuito de padronizar uma interface comum para comunicação de dados entre equipamentos. É utilizada largamente ainda hoje e tem importância na automação, pois grande parte dos equipamentos utilizam essa tecnologia para comunicação, seja através do padrão RS-232 ou muitos outros que transmitem seus dados serialmente como RS-485, *Ethernet* e USB.

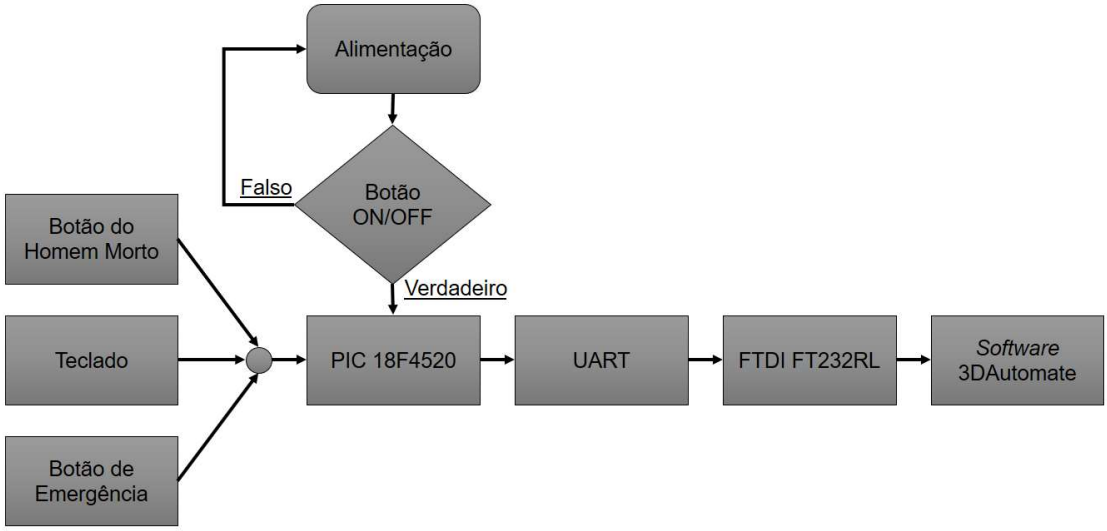
A comunicação é feita através da transmissão de informação bit a bit sequencialmente, normalmente caracteres ASCII através de uma única via (fio). Na grande maioria dos padrões existem vias de transmissão e recepção chamados de RX e TX. O padrão mais utilizado para transmissão serial nos microcontroladores é o RS-232, cujo controle da comunicação pode ser feito através da UART.

De acordo com SOUZA (2017), UART (*Universal Asynchrounous Receiver/Transmiter* ou Receptor/Transmissor Universal Assíncrono) é o periférico encontrado em alguns microcontroladores que possibilita a transmissão e a recepção de dados. O funcionamento desta é razoavelmente simples, os dados chegam em forma de palavras digitais que na prática são vários bits em paralelo (em um microcontrolador de 8 bits, o barramento é composto por 8 vias metálicas) e o periférico UART passa a transmitir esses bits sequencialmente por uma linha de dados, com um controle de paridade que garante a integridade dos dados.

3 MATERIAIS E MÉTODOS

Os principais componentes do projeto desenvolvido são: botões de contato, duas chaves tipo normalmente aberta e normalmente fechada, microcontrolador, chip para conversão da comunicação serial e um computador com um *software* robótico instalado. Na Figura 1, é possível ver o diagrama de blocos desse sistema.

Figura 1 – Diagrama de blocos *hardware* do kit



Fonte: Autoria própria

3.1 3DAutomate *Software*

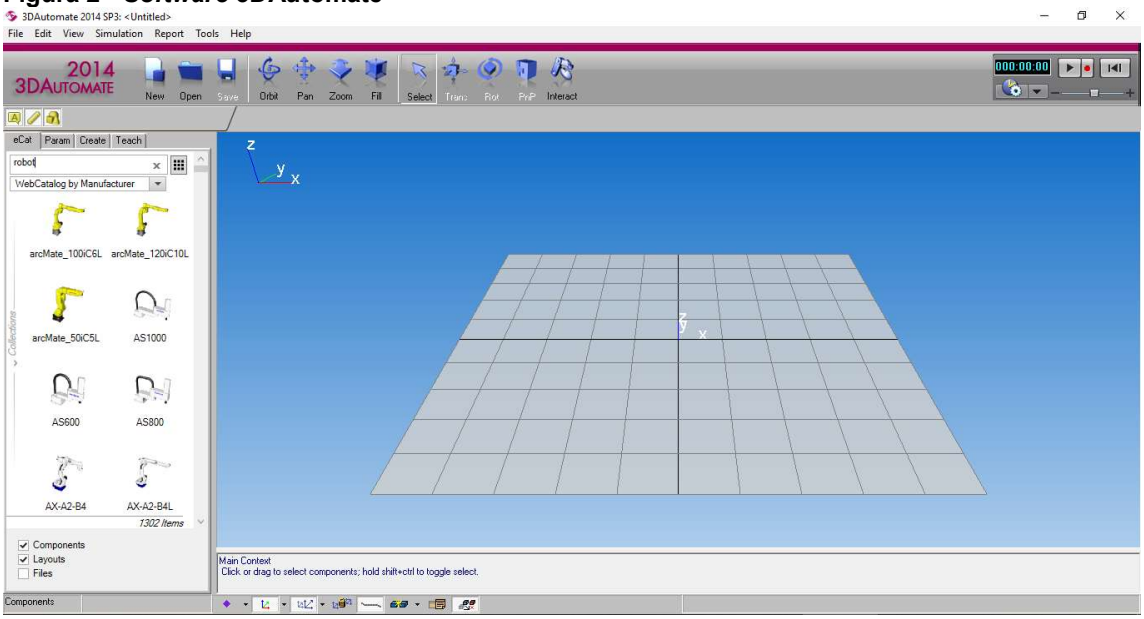
Uma empresa que disponibiliza e presta suporte para inúmeros *softwares* de robótica, é a *Visual Components*. Segundo WALTER (2016), uma empresa de origem finlandesa, especializada em designs avançados e simulações para linhas de produção; processos inteiros de manufatura podem ser simulados e analisados, incluindo componentes robóticos, fluxo de materiais, ações manuais de operadores, entre outros. Foi escolhido o *software* 3DAutomate da empresa *Visual Components* por ser largamente utilizado, conforme WALTER (2016) diz e porque esta empresa demonstrou completo apoio ao projeto disponibilizando gratuitamente o uso da plataforma.

De acordo com o site da companhia, 3DAutomate é um *software* de alta capacidade capaz de simular grandes linhas de produção, por isso ele é especialmente indicado para aplicações na indústria automobilística. Isso quer dizer que está preparado para lidar com simulações complicadas e longas, entregando

soluções livre de erros e de resposta rápida. No caso deste projeto, será realizada uma simulação simples: apenas um único robô com seis graus de liberdade. Porém, este robô será controlado externamente via comunicação serial e, para realizar tal feito, é necessário um *software* programável, ou seja, que permita modificações e confecções de novos projetos. 3DAutomate da *Visual Components*, é um *software* eficaz e programável em Python.

Na Figura 2 é possível observar a imagem do *software* aberto com algumas opções de projetos.

Figura 2 - *Software* 3DAutomate



Fonte: Autoria própria

3.1.1 Robô Fanuc LR Mate 200 iC

Na plataforma 3DAutomate existe algumas possibilidades de robôs manipuladores para escolher. Dentre estas, o robô da marca Fanuc LR Mate 200 iC. De acordo com ABDERRAHMANE (2014), um robô muito conhecido no meio e amplamente utilizado para fins educacionais em diversas universidades no mundo. Pelo alto número de material disponível sobre este robô, incluindo vídeos detalhando seu movimento e a familiaridade com o produto, o robô LR Mate 200 iC foi escolhido para dar prosseguimento ao projeto.

No EUROBOTS (2018), podemos encontrar algumas informações sobre o Fanuc LR Mate 200iC. É descrito como sendo um robô compacto, com 6 graus de liberdade, braço de alta velocidade e alta precisão de posicionamento, sendo ideal

para uma variedade de aplicações de automação; está equipado com o controlador R30iA moderno e versátil.

Com o intuito de descrever as relações entre os eixos das juntas, são usadas equações da cinemática robótica, determinando assim, a posição e orientação de um eixo 'i-1' em relação ao eixo 'i'. De acordo com ODEYINKA (2016), o método mais comum é utilizando a matriz de transformação homogênea desenvolvida por Denavit-Hartenberg, também conhecido como parâmetros D-H. Conforme CABRAL (2018), são quatro parâmetros associados a uma convenção para fixar sistemas de referência aos elos de um manipulador robótico. Para entender melhor o funcionamento do controlador e a maneira como o robô se movimenta, devemos começar analisando suas teorias cinemáticas através dos parâmetros D-H apresentados na Tabela 1.

Tabela 1 – Parâmetros D-H para o Fanuc LR Mate 200iC

Junta	d_i	θ_i	a_i	α_i
1	330	0°	75	-90°
2	0	-90°	300	180°
3	0	180°	-75	90°
4	-320	0°	0	-90°
5	0	0°	0	90°
6	-80	180°	0	180°

Fonte: DJURIC (2017)

A matriz de transformação homogênea ${A_i}^{i-1}$ expressa na Equação 1, mostra a transformação do link (corpo rígido do braço) para a i^n junta; onde $i = 1, 2, 3, \dots, n$ e n é o número de links. O modelo cinemático completo para o Fanuc LR Mate 200iC foi calculado e validado utilizando o *software* Matlab, baseando se nas equações da cinemática direta presentes nas Equações 2 à 7. A relação com os parâmetros D-H vistos na Tabela 1 para $i = 1, 2, 3, 4, 5, 6$ são mostrados nas Equações 8 e 9.

$$ {A_i}^{i-1} = \begin{bmatrix} cos\theta_i & -\cos\alpha_i \operatorname{sen}\theta_i & \operatorname{sen}\alpha_i \operatorname{sen}\theta_i & \alpha_i \cos\theta_i \\ sen\theta_i & \cos\alpha_i cos\theta_i & -\operatorname{sen}\alpha_i \cos\theta_i & \alpha_i \operatorname{sen}\theta_i \\ 0 & \operatorname{sen}\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

$$A_1{}^0 = \begin{bmatrix} cos\theta_1 & 0 & -\text{sen}\theta_1 & \alpha_i \cos\theta_1 \\ sen\theta_1 & 0 & \cos\theta_1 & \alpha_i \text{sen}\theta_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$A_2{}^1 = \begin{bmatrix} cos\theta_2 & -\text{sen}\theta_2 & 0 & \alpha_2 \cos\theta_2 \\ sen\theta_2 & -cos\theta_2 & 0 & \alpha_2 \text{sen}\theta_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$A_3{}^2 = \begin{bmatrix} cos\theta_3 & 0 & \text{sen}\theta_3 & \alpha_3 \cos\theta_3 \\ sen\theta_3 & 0 & -\cos\theta_3 & \alpha_3 \text{sen}\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$A_4{}^3 = \begin{bmatrix} cos\theta_4 & 0 & -\text{sen}\theta_4 & 0 \\ sen\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$A_5{}^4 = \begin{bmatrix} cos\theta_5 & 0 & \text{sen}\theta_5 & 0 \\ sen\theta_5 & 0 & -\cos\theta_5 & 0 \\ 0 & 1 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$A_6{}^5 = \begin{bmatrix} cos\theta_6 & -\text{sen}\theta_6 & 0 & 0 \\ sen\theta_6 & -cos\theta_6 & 0 & 0 \\ 0 & 0 & -1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$A_6{}^0 = A_1{}^0 + A_2{}^1 + A_3{}^2 + A_4{}^3 + A_5{}^4 + A_6{}^5 \quad (8)$$

Onde $A_6{}^0$ mostra a relação entre o fim de curso do robô e sua coordenada absoluta (posição das coordenadas de sua base).

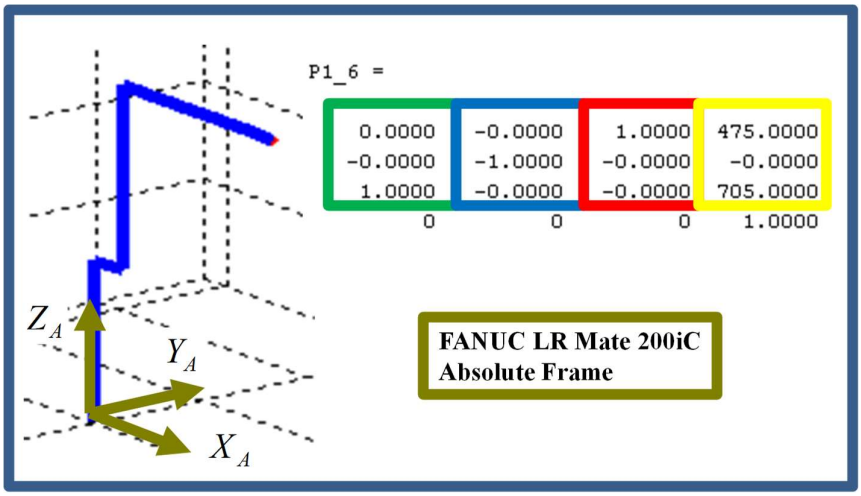
$$A_6{}^0 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

O valor numérico apresentado na Tabela 1 é usado para calcular as coordenadas da posição de fim de curso e orientação em relação à coordenada

absoluta. A matriz de transformação homogênea é mostrada na Equação 10 e na Figura 3.

$$A_6{}^0 = \begin{bmatrix} 0 & 0 & 1 & 475 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 705 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Figura 3 - Cinemática do robô Fanuc LR Mate 200 iC em relação à sua coordenada absoluta



Fonte: DJURIC (2017)

A validação do Fanuc LR Mate 200iC foi feito baseado na coordenada de mundo do robô, onde temos $d_1 = 330mm$ acima do eixo absoluto (base) z. Provando que a Equação 10 traz o resultado correto para a equação cinemática deste robô. É utilizando estes valores e equações, que o controlador é capaz de determinar o movimento do braço robótico de maneira otimizada.

O robô é constituído então pelo controlador R30iA, o braço robótico responsável por realizar os movimentos e uma ferramenta se faz necessária para garantir a comunicação entre o operador e o robô. Esta ferramenta recebe o nome de *Teach Pendant*. Na Figura 4 é apresentada uma imagem do conjunto robô Fanuc LR Mate 200iC, onde podemos ver o braço robótico, o controlador R30iA e o *Teach Pendant*.

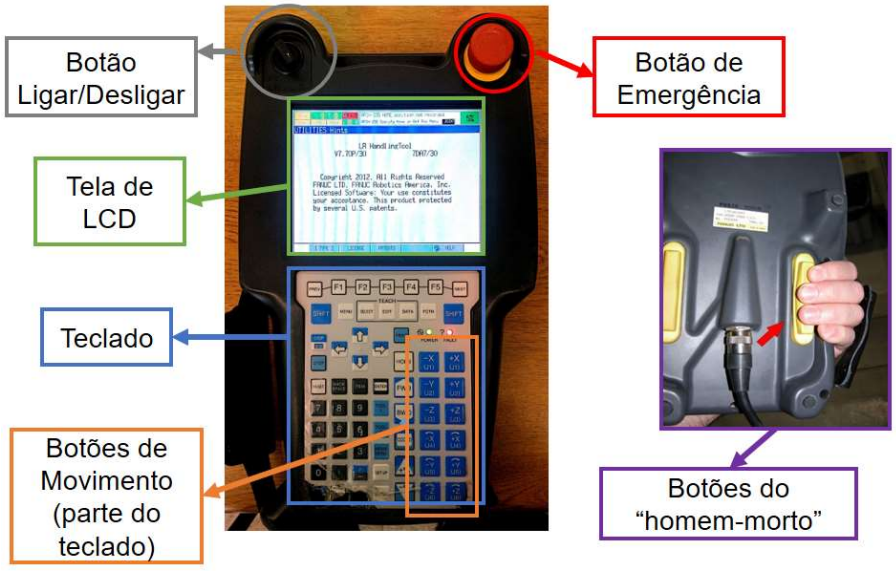
Figura 4 – Robô Fanuc LR Mate 200iC

Fonte: EUROBOTS (2018)

Como o intuito deste trabalho é auxiliar no aprendizado de robótica e treinar pessoal para utilizar o sistema robótico, o foco será reproduzir o *Teach Pendant*. Visto que o braço robótico é provido pelo *software* de simulação 3DAutomate, a comunicação com o controlador R30iA torna-se desnecessária.

O *Teach Pendant* consiste basicamente de 6 elementos como mostra a Figura 5.

Figura 5 – Elementos do Teach Pendant

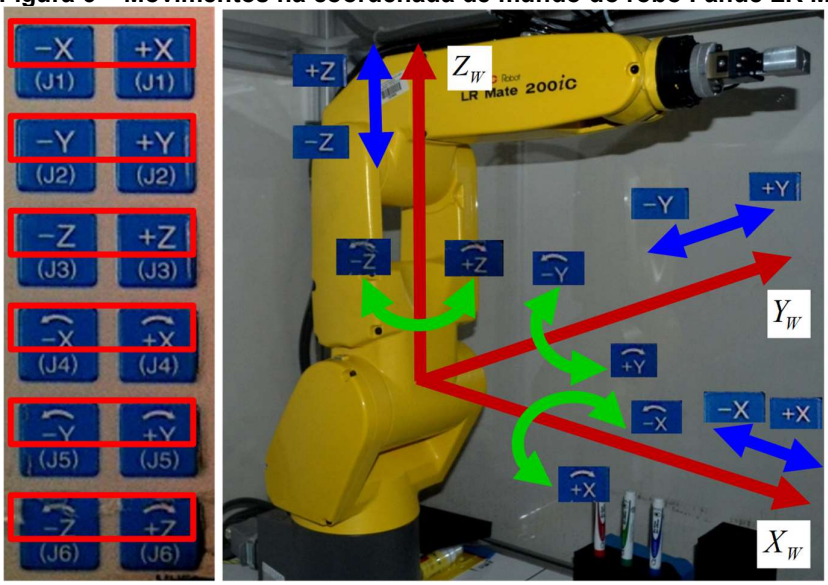


Fonte: Autoria própria

Nas Figuras 6 e 7 é possível observar que são 12 botões responsáveis pelo movimento do robô, sendo que os símbolos na parte superior são sobre o movimento

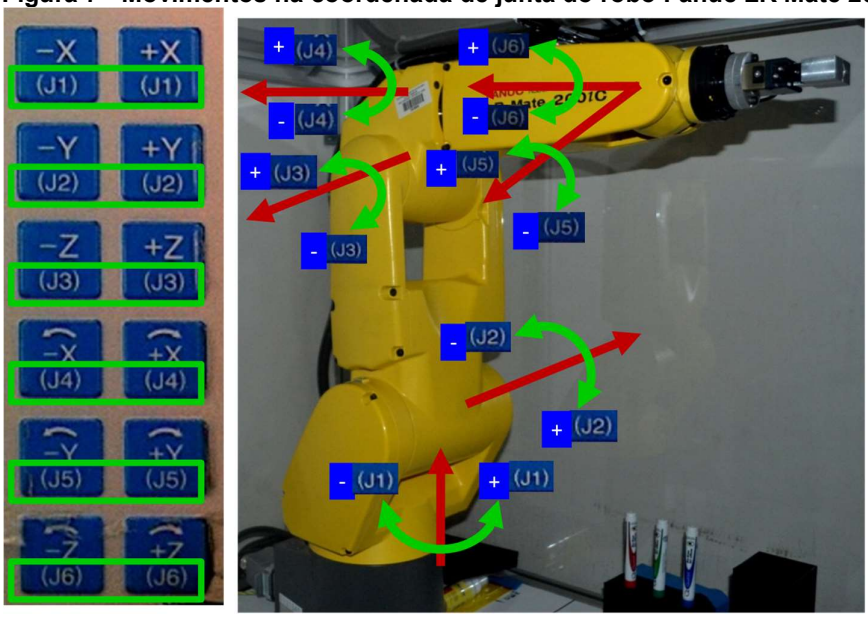
do robô na coordenada de mundo e os símbolos da parte inferior, na coordenada de junta. Quando falamos de movimento na coordenada de mundo, são possíveis 3 translações nos eixos X, Y e Z e 3 rotações nestes mesmos eixos. Na coordenada de junta, são 2 rotações (positiva ou negativa) para cada junta, totalizando 6 rotações.

Figura 6 – Movimentos na coordenada de mundo do robô Fanuc LR Mate 200 iC



Fonte: Autoria própria

Figura 7 - Movimentos na coordenada de junta do robô Fanuc LR Mate 200 iC



Fonte: Autoria própria

3.2 Microcontrolador PIC18F4520

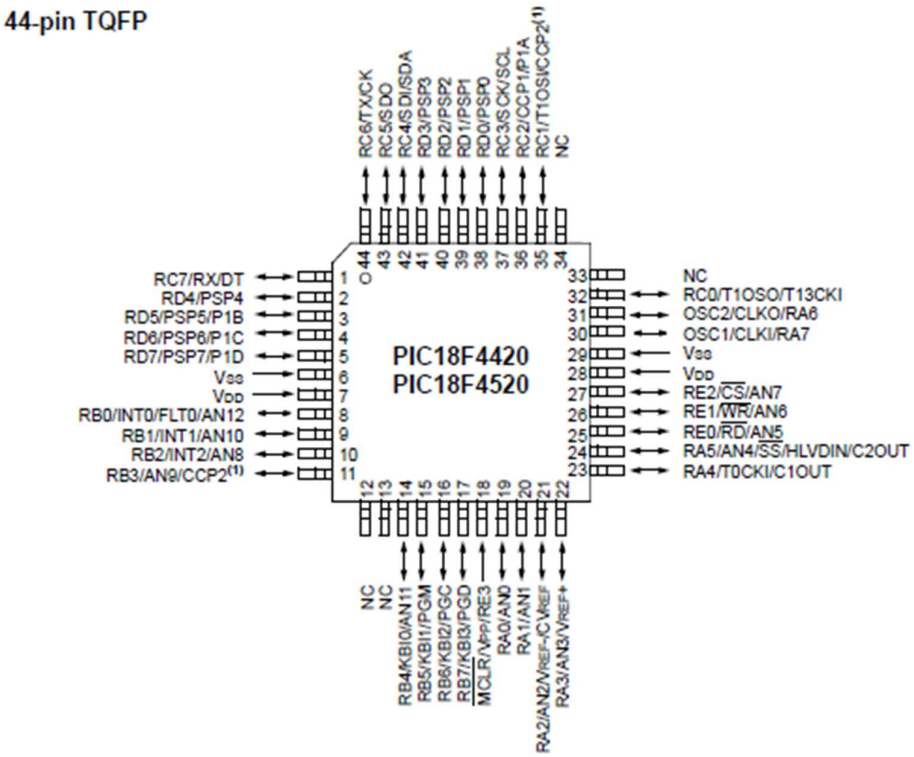
A escolha do microcontrolador para este projeto, foi tomada com base principalmente nos periféricos necessários. Como foi reproduzido um teclado grande com 63 teclas e havia a motivação de deixar o circuito preparado para a futura instalação de uma teca LCD, era obrigatório que o microcontrolador disponibilizasse pelo menos 31 portas de I/O. Além disso, era essencial a existência de um periférico para comunicação com o computador. Também foi analisado a necessidade deste microcontrolador possui uma memória interna grande o suficiente para a gravação do código fonte. Por estes motivos e pela familiaridade com os componentes, a escolha foi entre o PIC 18F4520 e o PIC 18F4550, ambos da Microchip. De maneira geral os 2 microcontroladores são bem parecidos, com a diferença do 18F4550 possuir comunicação USB, indisponível no 4520. Porém, havia disponível para uso o PIC 18F4520 que atende as necessidades do projeto.

A Microchip é uma empresa produtora de diversos microcontroladores e que possui em seu portfólio também circuitos analógicos e digitais, além de memórias e diferentes sensores. Segundo SOUZA (2017) seus produtos são muito utilizados aqui no Brasil, o que a torna uma das principais fornecedoras para o mercado brasileiro.

De acordo com MICROCHIP Technology Inc. (2017, B), a empresa é uma fornecedora líder de microcontroladores e semicondutores analógicos, que oferece o desenvolvimento de produtos com baixo risco, menor custo total e tempo de comercialização mais rápido para milhares de aplicações de clientes diferentes em todo o mundo. Sediada em Chandler, Arizona, a Microchip oferece excelente suporte técnico, juntamente com entrega confiável e alta qualidade.

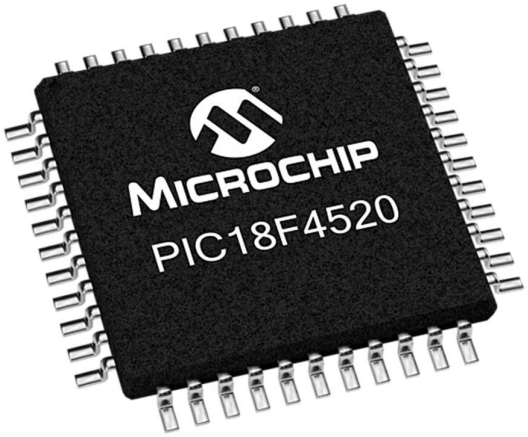
O Microcontrolador PIC18F4520 oferece um conjunto de instruções e funcionalidades bastante amplo e possui encapsulamento PDIP 44 pinos, contém 256 bytes de memória EEPROM e 32Kb de memória Flash. Neste projeto foi utilizado o PIC18F4520-I/PT, SMD, TQFP de 44 pinos; é mostrado na Figura 8 o diagrama de pinos deste microcontrolador, retirado do *datasheet*, enquanto na Figura 9 pode ser vista uma foto deste microcontrolador.

Figura 8 – Diagrama de pinos PIC 18F4520



Fonte: MICROCHIP (2016)

Figura 9 - Foto do microcontrolador PIC 18F4520



Fonte: MICROCHIP (2017, A)

Este microcontrolador apresenta diversos periféricos avançados como: UART, CAN, Ethernet, LCD entre outros. Suas principais características são: 83 instruções com otimização para programação em linguagem C; até 2 MB memória de programa; 4KB de memória RAM; pilha de *hardware* de 32 níveis; multiplicador 8x8 por *hardware*; melhor performance dos microcontroladores de 8 bits da Microchip.

3.3 Chip FT 232 RL

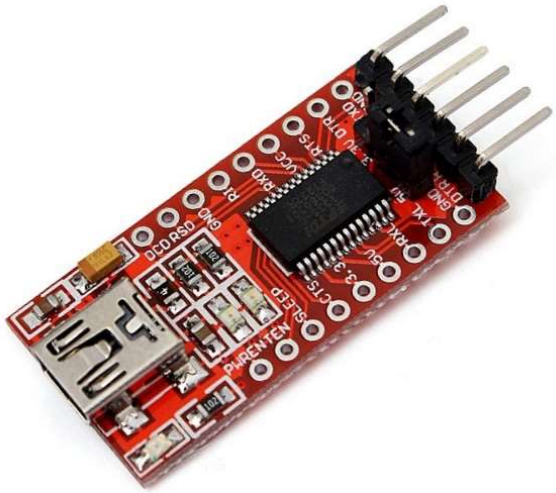
Para o funcionamento do projeto, o microcontrolador deve se comunicar com o *3DAutomate* instalado em um computador. Pela baixa complexidade de desenvolvimento e baixo custo, foi escolhida a comunicação serial baseada no protocolo RS-232, presente no PIC 18F4520.

No entanto, embora sejam encontradas em computadores atuais, as portas seriais baseadas no protocolo RS-232 vêm sendo gradativamente substituídas pelas portas de comunicação USB, principalmente em notebooks. Desta forma, para realizar a comunicação com o computador, o dispositivo projetado fará uso de um conversor USB-RS232 para se ligar fisicamente a uma porta USB e fazer uso de uma COM virtual para transmitir os dados pela UART do microcontrolador.

Foi escolhido utilizar o conversor da FTDI, o FT232RL, que é capaz de realizar esta conversão de maneira eficiente. Além da facilidade de utilização e das ótimas funcionalidades, este chip também é compatível com qualquer computador, necessitando apenas da instalação de um *drive* disponível no site do fabricante para uso.

Além disso, optou-se pela compra de um módulo completo para a utilização por dois motivos: maior facilidade para comprar o componente e menor chance de encontrar problemas na hora de testar a comunicação. Na Figura 10 é possível ver uma foto do circuito adquirido.

Figura 10 – Placa FT 232RL



Fonte: FILIPEFLOP (2018)

3.4 Alimentação

Tem-se dois circuitos independentes no kit: um contendo o microcontrolador e outro sendo o módulo do FT 232 RL. O módulo é alimentado diretamente pela porta USB na qual está conectado, com uma tensão de 5V, utilizando a conexão para transferência de dados. O microcontrolador PIC 18F4520, de acordo com seu *datasheet*, necessita de uma tensão entre 2 e 5,5V.

Outro ponto considerado, é a preocupação em deixar o projeto preparado para no futuro, ser adicionado uma tela LCD que também precisa ser alimentada. Sendo assim, foi decidido por utilizar uma fonte externa de 5V e 2A.

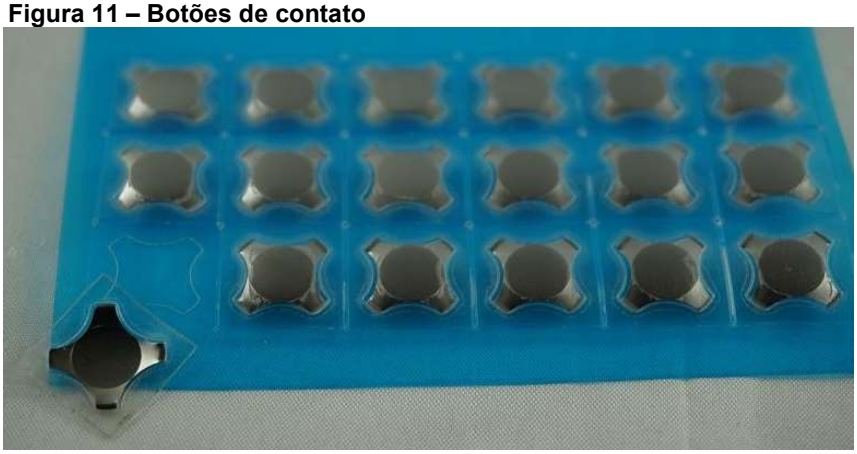
3.5 Componentes Auxiliares

Outros componentes necessários para o funcionamento do kit são:

3.5.1 Botões de Contato

O teclado é formado por botões de contato simples, pela possibilidade de utilizar uma foto do *Teach Pendant* e ficar o mais parecido possível com a realidade. Na Figura 11 há uma fotografia dos botões utilizados.

Figura 11 – Botões de contato



Fonte: ALIEXPRESS (2017, A)

3.5.2 Botão do Homem Morto

Uma parte importante do *Teach Pendant*, são os botões do homem morto (*Dead Man Button*). Recebem este nome pelo seu método de funcionamento: permitem a utilização do teclado apenas quando pelo menos um destes botões, que ficam na parte traseira do *Teach,* está pressionado; ou seja, a pessoa que está operando está consciente de suas ações. No caso de um acidente e o operador estar em perigo, ele não será capaz de manter o botão pressionado e o robô irá parar o movimento instantaneamente, evitando maiores problemas. Na Figura 12 uma foto deste botão.

Figura 12 – Botão do homem morto Fanuc LR Mate 200 iC

Fonte: Autoria própria

Infelizmente, não foi possível encontrar estes botões para compra, mas com o intuito de manter a funcionalidade parecida com o verdadeiro *Teach Pendant*, foram adicionados um botão de contado de cada lado do teclado e incluído o mesmo método de funcionamento na execução do código.

3.5.3 Botão de Emergência

Há também na parte superior do *Teach Pendant* um botão de emergência vermelho do tipo cogumelo. Quando pressionado, ele para o movimento do robô até ser solto novamente, ou seja, mesmo funcionamento de quando o botão do homem morto é solto. É demonstrado na Figura 13 o botão comprado para este propósito.

Figura 13 – Botão de Emergência vermelho do tipo cogumelo

Fonte: ALIEXPRESS (2017, B)

3.5.4 Botão Liga/Desliga

O último periférico necessário, é um botão do tipo normalmente aberto para funcionar como uma chave liga/desliga para o kit. Foi adquirido um modelo parecido com o do *Teach Pendant* original, como mostra a Figura 14.

Figura 14 – Botão liga/desliga

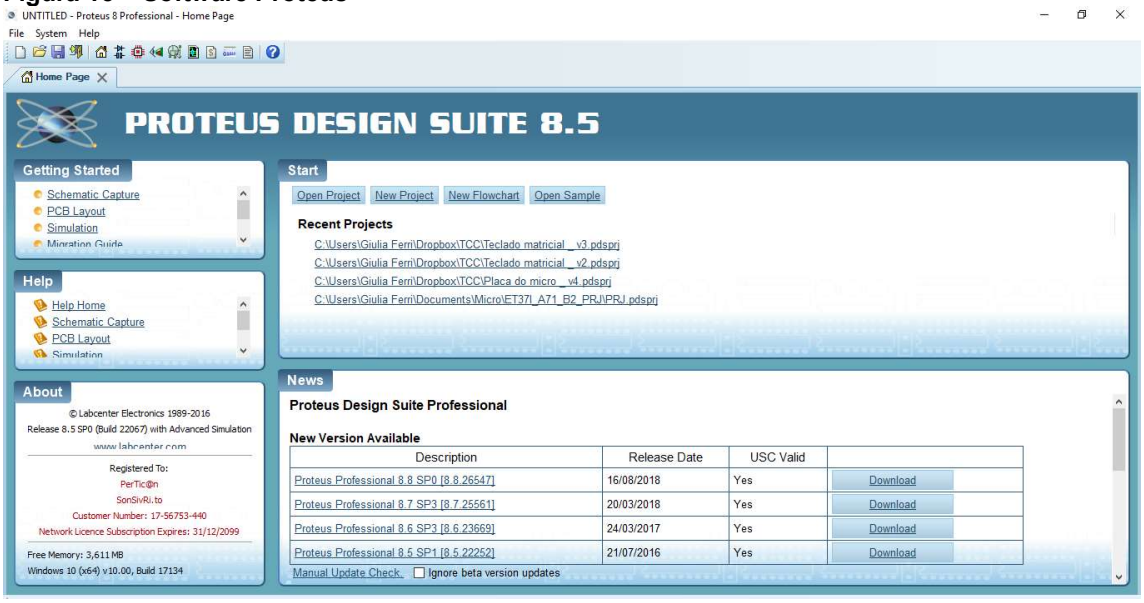
Fonte: ALIEXPRESS (2017, C)

3.6 *Softwares*

3.6.1 Proteus

Para o desenvolvimento das placas de circuito impresso, foi utilizado o *software* Proteus. De acordo com PROTEUS (2017), é um *software* completo para simulação e confecção de PCB (*Printed Circuit Board* – Placa de Circuito Impresso), de forma simples e poderosa. Contém vários módulos e ferramentas, assim como inúmeros componentes já integrados, o que facilita a produção de qualquer sistema embarcado. A Figura 15 traz uma captura de tela deste ambiente em funcionamento.

Figura 15 – Software Proteus

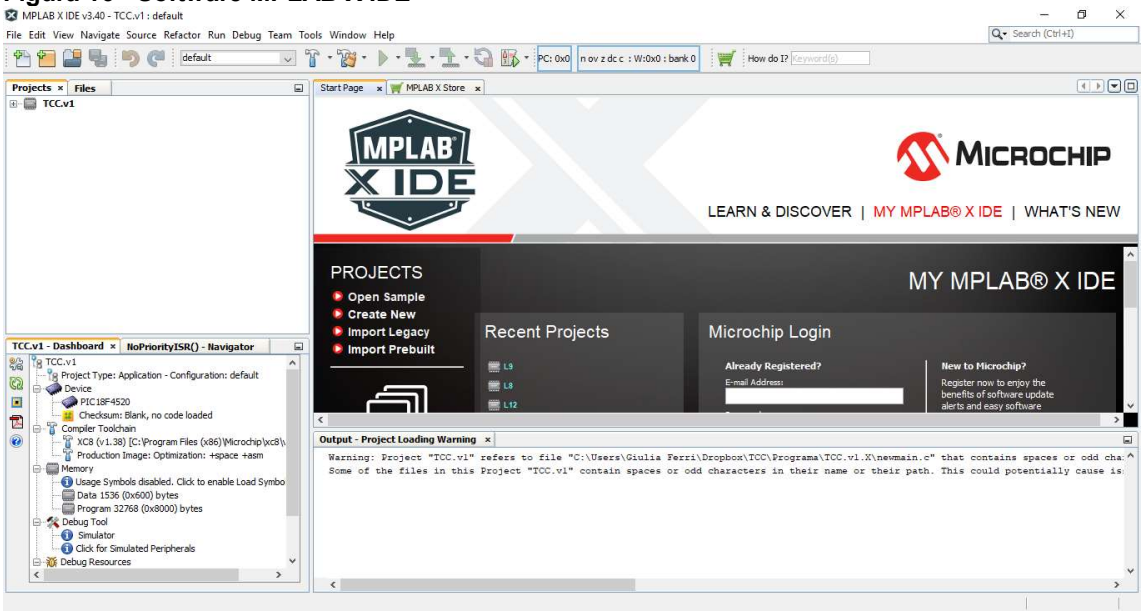


Fonte: Autoria própria

3.6.2 MPLAB X IDE

O microcontrolador foi programado em linguagem C na plataforma MPLAB X IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado). Um *software* fornecido gratuitamente pela Microchip, para editar e desenvolver sistemas embarcados que utilizam os microprocessadores da empresa, permite inclusive simulação e gravação dos microcontroladores. Na Figura 16 é possível ver este *software* aberto com as opções de criar projetos.

Figura 16 - Software MPLAB X IDE



Fonte: Autoria própria

4 DESENVOLVIMENTO

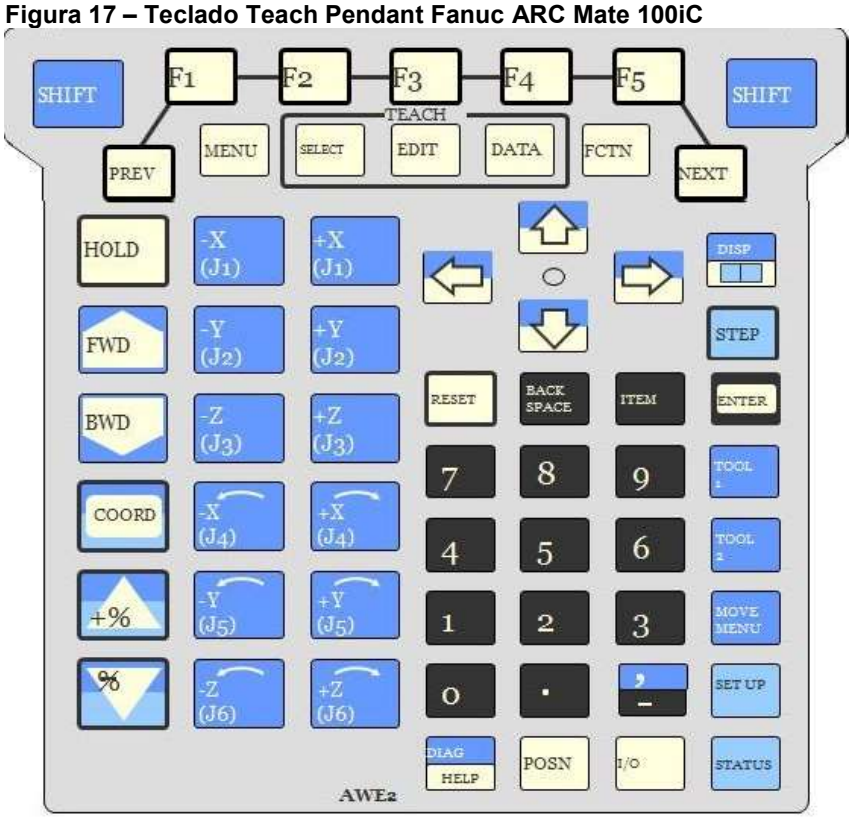
Para o desenvolvimento do kit, há 2 vieses separados: a estrutura física, contendo as placas e componentes; e a estrutura computacional, com os programas e *softwares* necessários para garantir o movimento do robô.

4.1 Estrutura Física

4.1.1 Teclado

O teclado do kit, foi desenvolvido para ser parecido com o teclado do *Teach Pendant* real, porém com o dever de ser funcional e executável. Além disso, é necessária uma imagem do teclado que possua alta qualidade para ser impressa e colada sobre a placa, com esse intuito foi utilizada a imagem da Figura 17 como base, o teclado do Fanuc ARC Mate 100iC.

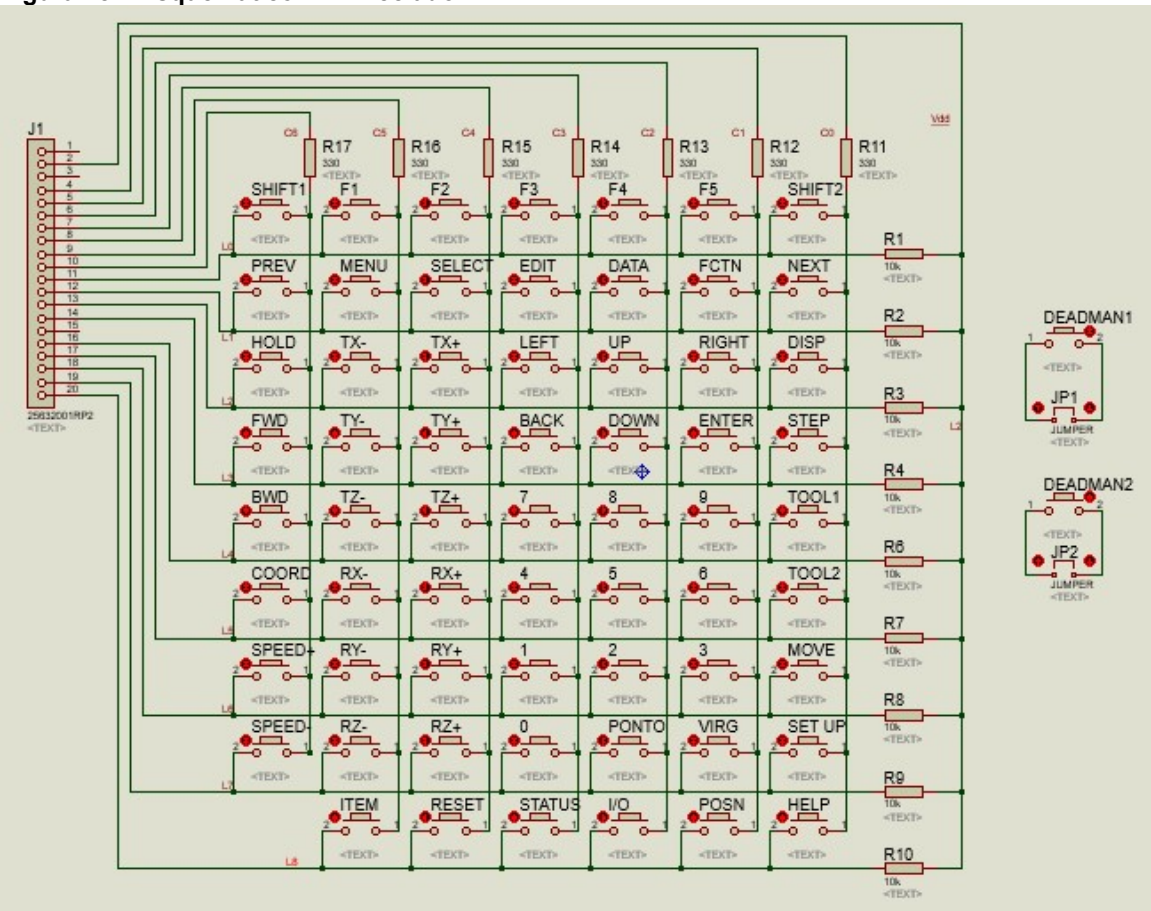
Figura 17 – Teclado Teach Pendant Fanuc ARC Mate 100iC



Fonte: Adaptado FANUC (2018)

Com estas informações, foi definido o esquemático da PCB do teclado, visível na Figura 18.

Figura 18 – Esquemático PCB Teclado



Fonte: Autoria própria

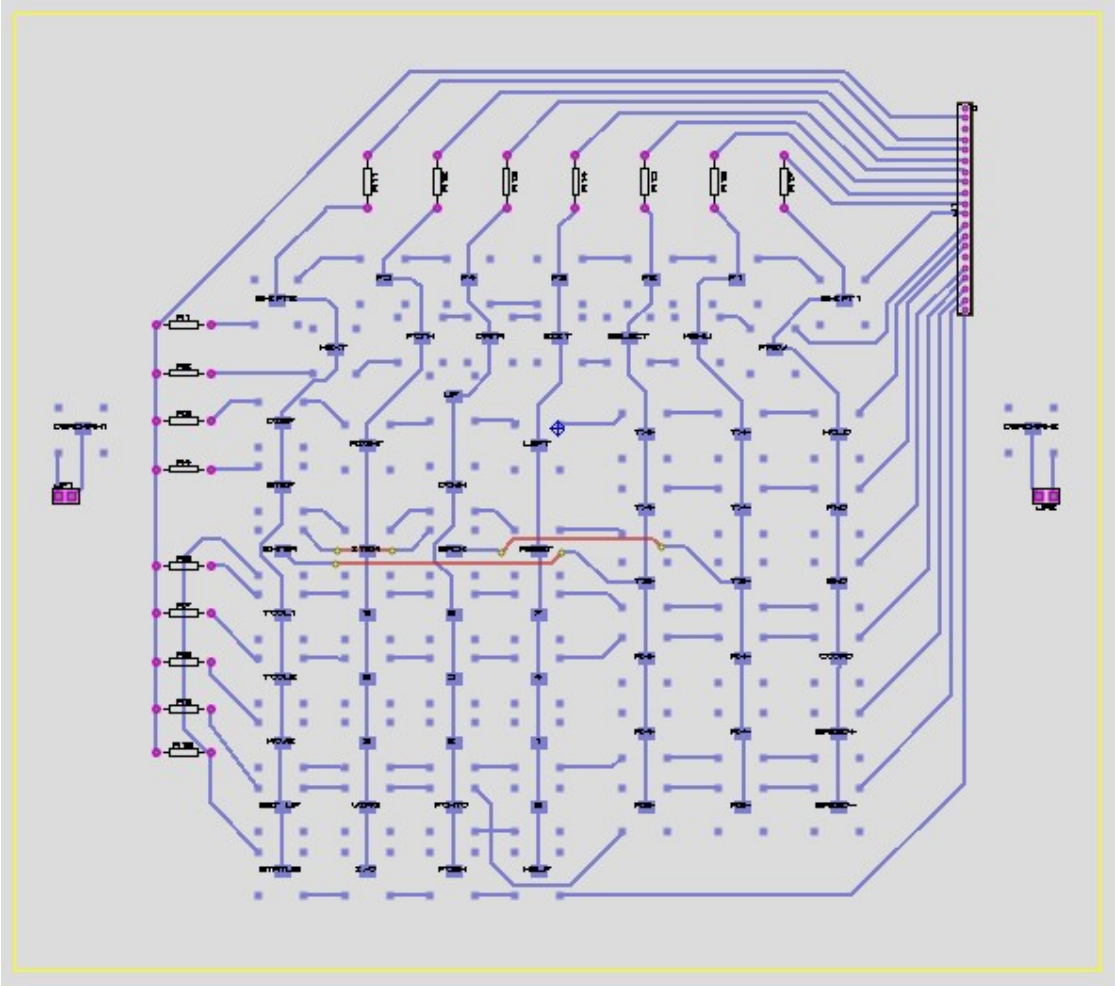
Como há 62 botões à serem analisados, foi necessário implementar um sistema de varredura na qual o teclado foi dividido em 9 linhas e 7 colunas e em cada cruzamento de uma linha e uma coluna há um botão. Além disso, há uma ligação em comum de todos os botões à uma porta setada (nível lógico alto). Desta maneira, são utilizadas apenas 17 portas de I/O para verificar o teclado inteiro.

A varredura deste teclado é feita por *polling,* onde uma coluna por vez é colocada em nível logico alto e o microcontrolador, checa por nível alto de resposta nas linhas. Quando é enviado o nível alto via coluna e recebido como resposta via linha, sabe-se que o botão alocado naquele cruzamento está pressionado.

Houve uma preocupação em adicionar os botões com os nomes corretos para facilitar a identificação no momento da programação, além de ter os botões do

homem morto (DEADMAN1 e DEADMAN2) na placa do teclado, ou seja, ao alcance do usuário. Na Figura 19, vemos o *layout* desta placa pronto para impressão.

Figura 19 – Layout PCB Teclado



Fonte: Autoria própria

Na Figura 20, uma foto desta placa finalizada com os botões e os outros componentes já soldados.

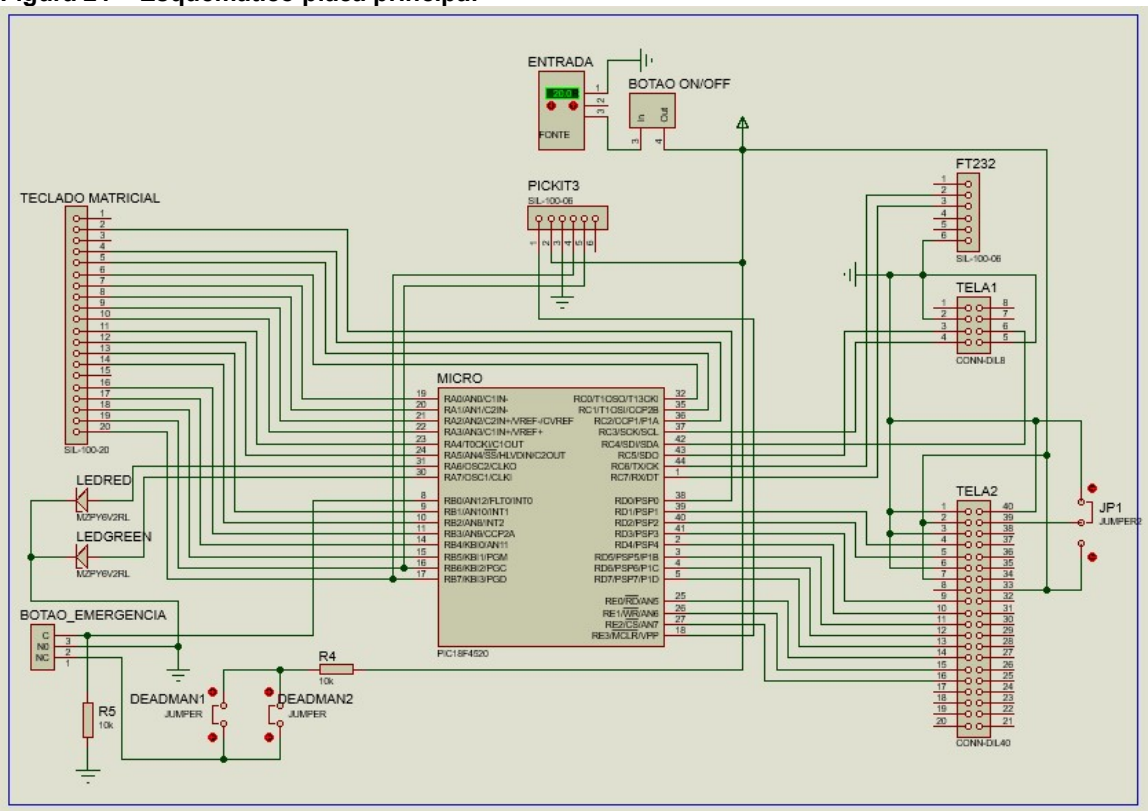
Figura 20 – Placa do teclado finalizada

Fonte: Autoria própria

4.1.2 Placa Principal

A placa principal deste projeto, contém o microcontrolador, o módulo FT 232 RL e todos os periféricos necessários. Conforme mostra o esquemático na Figura 21.

Figura 21 – Esquemático placa principal



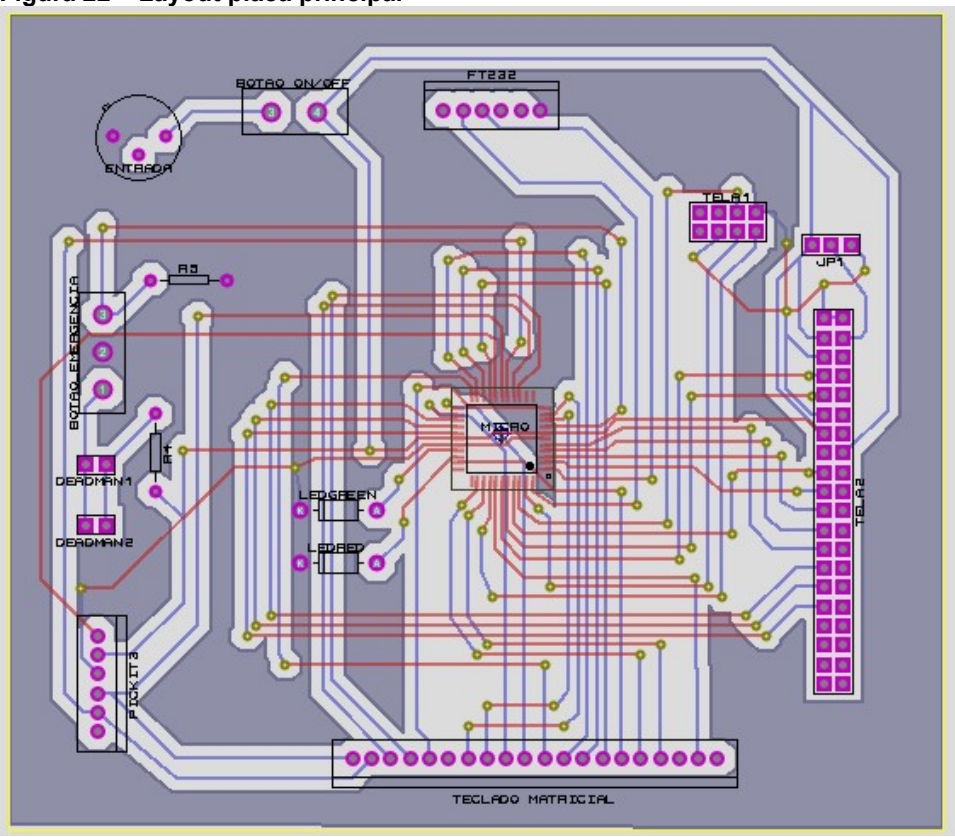
Fonte: Autoria própria

Além dos elementos já mencionados, há também alguns adicionais: 2 LEDs para indicar funcionamento adequado (verde) ou emergência (vermelho); barramento de 6 pinos para gravação do microcontrolador utilizando o gravador da Microchip, PICkit3; 2 barramentos de 40 e 8 pinos para a instalação de uma tela LCD em um trabalho futuro.

O PICkit3 é um gravador de baixo custo, mais rápido se comparado com versões anteriores e que realiza a gravação dos microcontroladores PIC por meio de conexão ICSP (*In Circuit Serial Programming*). Possui também a função de *debug* (depurador), onde você pode conectar o Pickit 3 diretamente no circuito e verificar em tempo real as condições e configurações do equipamento.

O *layout* da placa para impressão, pode ser visto na Figura 22.

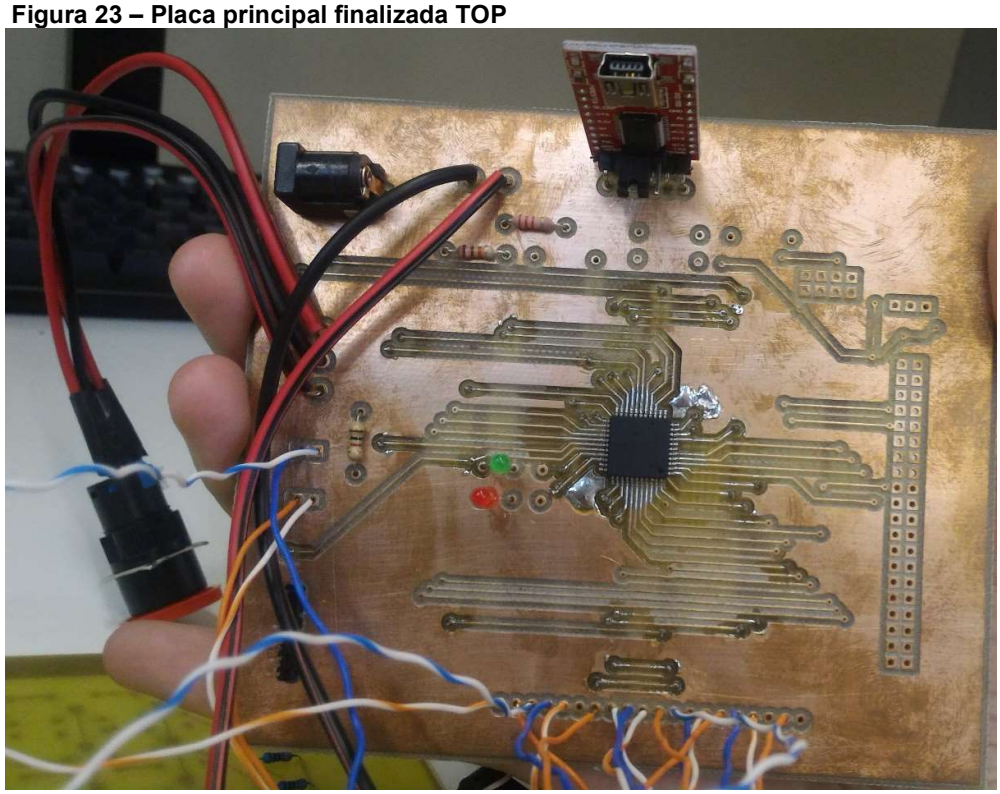
Figura 22 – Layout placa principal



Fonte: Autoria própria

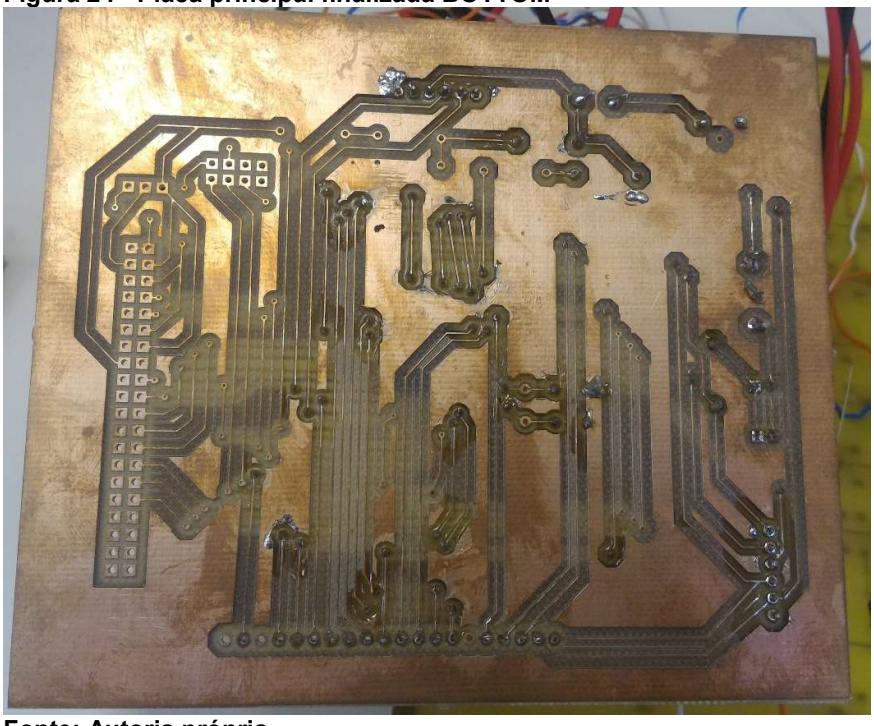
Esta é uma placa dupla face, ou seja, existem conexões tanto na face de cima (TOP) quanto na face de baixo (BOTTOM). As fotos dela já finalizada, estão nas Figuras 23 e 24.

Figura 23 – Placa principal finalizada TOP



Fonte: Autoria própria

Figura 24 - Placa principal finalizada BOTTOM



Fonte: Autoria própria

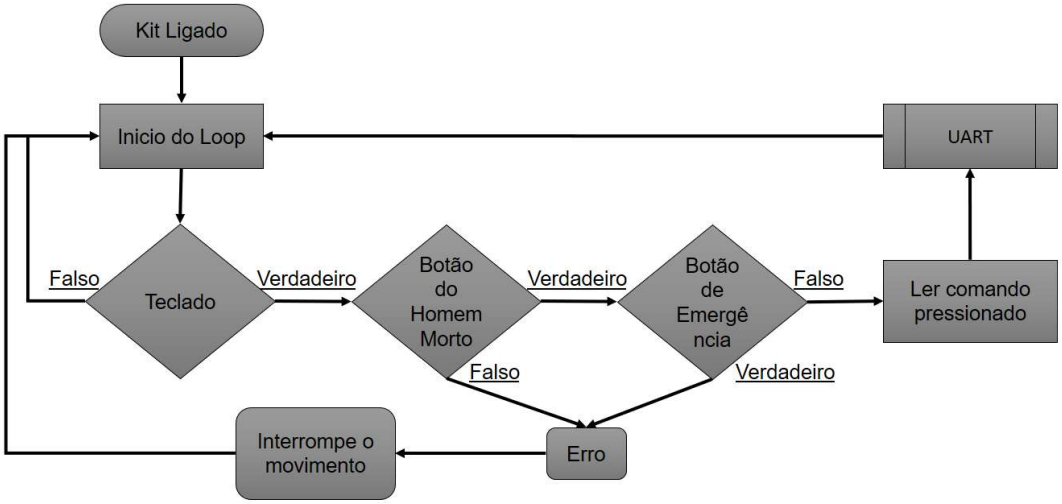
4.2 Estrutura computacional

4.2.1 Programação microcontrolador

A função do microcontrolador neste sistema, pode ser resumido da seguinte maneira: realizar a leitura do teclado, enviar via comunicação serial qual botão está sendo pressionado e respeitar as rotinas de segurança impostas pelo botão do homem-morto e pelo botão de emergência.

Na Figura 25, há um diagrama de blocos que mostra o funcionamento do programa desenvolvido.

Figura 25 – Diagrama de blocos do software (no PIC)



Fonte: Autoria própria

Observando a Figura 25, vemos que quando o sistema é ligado, o microcontrolador começa a verificar o teclado procurando por teclas pressionadas. Além disso, quando é utilizado teclas em um sistema embarcado é importante se preocupar com o efeito *boucing* das teclas. Este efeito aparece quando no momento de pressionar ou soltar uma tecla, acontece um contato momentâneo indesejado que o sistema reconhece como botão pressionado e realiza a ação para aquele botão. Em ordem de evitar isso, foi utilizada uma técnica conhecida por *de-bouncing*, onde é utilizado *delays* no código durante o período de instabilidade dos botões e assim evitar que os acontecimentos de *boucing* sejam detectados.

Quando encontrado um botão pressionado, o microcontrolador salva qual foi a tecla e verifica se o botão do homem morto está também pressionado e o botão de emergência solto. Caso afirmativo, uma mensagem é enviada via comunicação

serial para o computador, com o nome do botão. Se as rotinas de segurança não foram respeitadas, é indicado um sinal de erro para o PC que deve parar o movimento em ação e aguardar novas instruções.

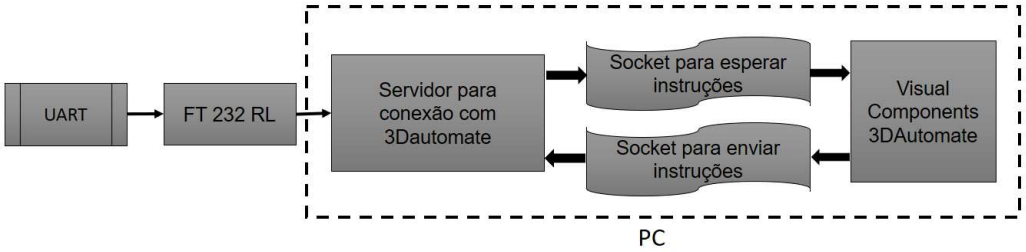
Como já mencionado anteriormente, foi utilizada a plataforma MPLAB X para escrita do código em C. O código pronto, pode ser verificado no Anexo A.

4.2.2 Programação 3Dautomate

Conforme explicado, as informações são enviadas do microcontrolador para o computador via comunicação serial. Porém, o *software* 3DAutomate não é capaz de realizar este tipo de comunicação. Sendo assim, um servidor foi criado no computador utilizando linguagem Python. O servidor cria *sockets* (interface de comunicação entre processos) para se comunicar com o *software* 3DAutomate; *sockets* são baseadas no Protocolo de Internet (*Internet Protocol* – IP), utilizando assim do endereço IP do computador para realizar a comunicação.

Este servidor é responsável então por receber os dados do microcontrolador via comunicação serial e enviar para o 3DAutomate pelos *sockets* criados. O programa é iniciado importando as bibliotecas para comunicação serial e para criar *sockets*, são criadas ordens para ler a entrada do computador e escrever para o 3DAutomate quando há informações; isto ocorre em um loop infinito. A Figura 26 traz o diagrama de blocos deste processo e o código para visualização está em Anexo B.

Figura 26 - Diagrama de blocos do software (no computador - PC)



Fonte: Autoria própria

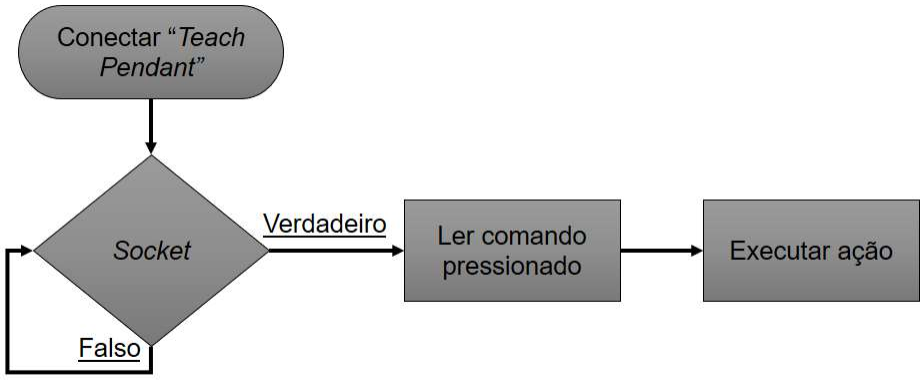
Foi mencionado que no *software 3DAutomate* foram feitas modificações no passado para que este se comunicasse com duas versões virtuais do *Teach Pendant*: uma na tela do computador e outra em aplicativo para celulares. Estas modificações foram resultado de uma pesquisa integrada entre a empresa *Visual Components* e a

Wayne State Univeristy. Contou com o apoio de um dos desenvolvedores do *3DAutomate,* Scott Walter e a Professora Doutora Ana M. Djuric. Assim, já havia um projeto onde as informações recebidas pela *socket* eram lidas e o robô se movimentava, porém só as 6 teclas de movimento tinham função.

No arquivo já existente, foram incluídas alterações para receber e escrever na *window chat* (janela de conversação entre o *software* e o usuário) quando uma tecla é pressionada. Além de colocar em funcionamento o botão de velocidade e o de coordenada, permitindo ao usuário alterar o quão rápido o braço se movimenta e se o movimento é realizado na coordenada mundo ou de junta. A necessidade de respeitar as rotinas de segurança já foi imposta pelo microcontrolador.

Como mostra o diagrama da Figura 27, o *3DAutomate* recebe as informações pela *socket* e reconhece qual comando foi pressionado, escrevendo este valor na *window chat.* Logo em seguida, é feito o tratamento da informação: se o botão pressionado for de movimento, o robô se move na direção desejada; quando o comando recebido é para diminuir ou aumentar a velocidade, é alterado o quão rápido o braço se mexe; e se o operador apertou o botão de coordenada, altera-se o plano em qual ocorre o movimento.

Figura 27 - Diagrama de blocos código no Software 3DAutomate



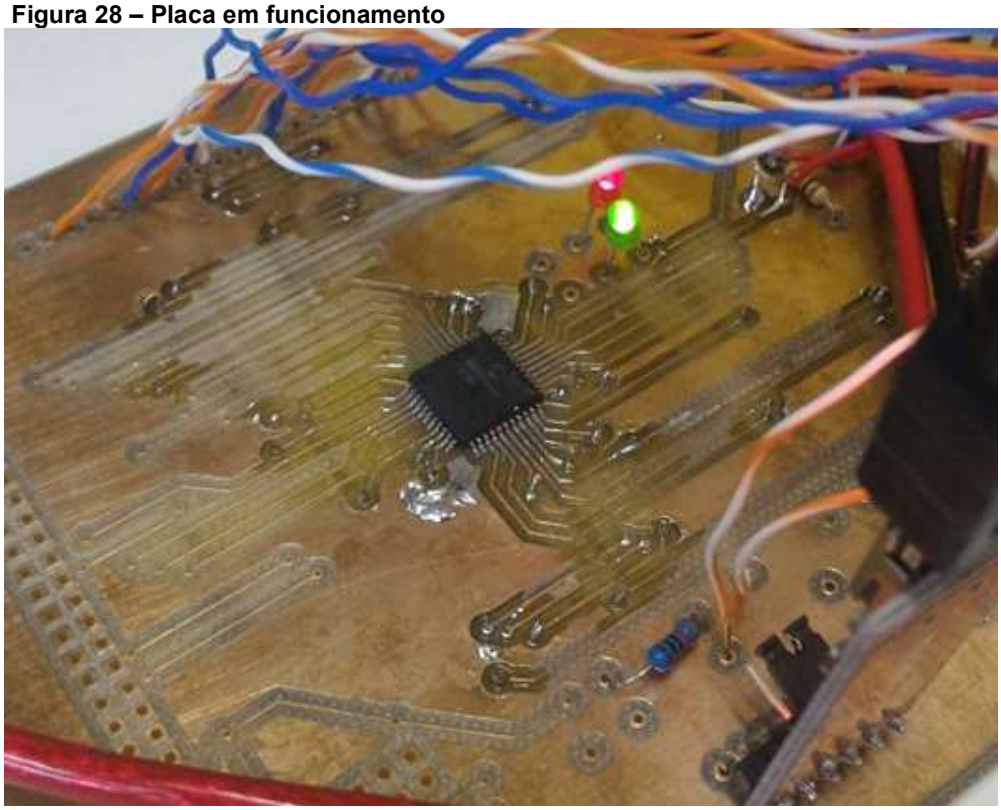
Fonte: Autoria própria

5 RESULTADOS OBTIDOS

Para atingir o objetivo, foi necessária a confecção de 2 placas de circuito impresso que foram desenhadas desde o esquemático até obter o *layout* final para impressão. Também foram desenvolvidos 3 códigos, um para o *firmware* do microcontrolador, outro para criar o servidor de comunicação e o último para compatibilizar este servidor com o *software 3DAutomate*.

Finalizando as etapas de desenvolvimento, iniciou-se os testes no kit para garantir o funcionamento conforme esperado. Foi verificado que quando conectado à porta USB do computador e ligado na energia, o PIC entra em atividade, o que é indicado pelos LEDs adicionados na placa (piscam 2 vezes). Isto mostra que o circuito funciona, a placa foi confeccionada corretamente e o microcontrolador foi programado. Na Figura 28 uma foto mostrando o fato descrito acima.

Figura 28 – Placa em funcionamento



Fonte: Autoria própria

O segundo passo, foi testar a ação da tecla quando pressionada. Conforme esperado, se respeitados os protocolos de segurança, aparece no computador uma mensagem informando qual botão foi apertado. Quando o botão do homem morto não

está pressionado ou o botão de emergência foi ativado, aparece a mensagem erro na tela.

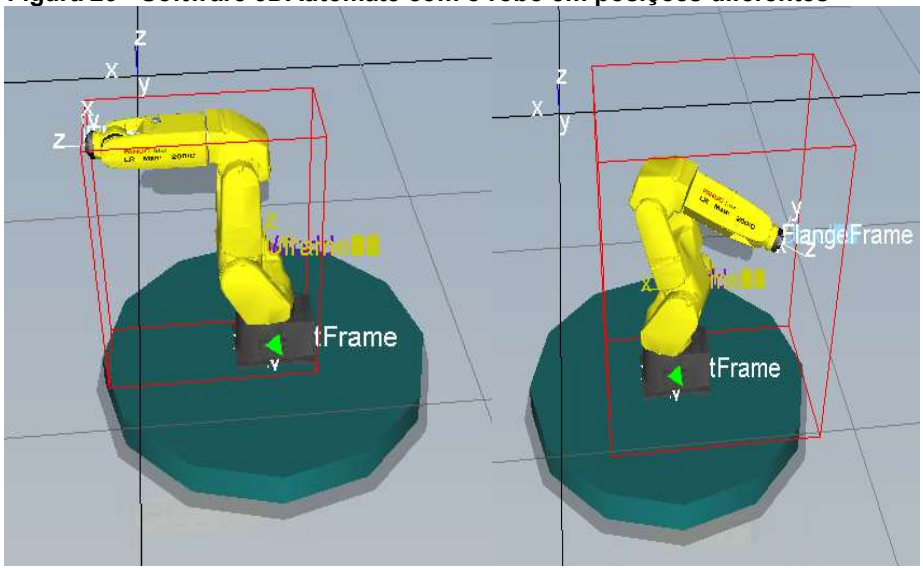
Contudo, os botões do teclado apresentaram inconsistência mecânica durante a utilização, mesmo utilizando a técnica *de-bouncer*, causando reações inesperadas na utilização do kit. Acredita-se que este problema esteja relacionado ao fato da solda não ter tido uma boa aderência ao botão, porém, não foi possível determinar a causa exata desta falha.

Embora funcional, as placas do kit foram conectadas por fios diretamente soldados. Um ponto de melhoria observado seria a implementação da conexão das placas do kit através de cabos tipo flat, o que permite maior mantenabilidade das placas no futuro.

Também foram considerados neste trabalho os pinos de GPIO (*General Pupose Input/Output)* necessários para comunicação paralela de 8 bits com um LCD mais pinos de controle, no entanto esta funcionalidade não foi implementada neste projeto.

O último teste, foi o funcionamento do *3DAutomate* em conjunto com o kit. O *software* indica qual tecla foi pressionada e quando solicitado, realiza os movimentos do braço. Além de alterar a velocidade de movimento para cima ou para baixo e a coordenada de movimento conforme a necessidade do usuário. Na Figura 29, imagens da tela com o robô em 2 posições, antes e depois de ser movimentado pelo kit.

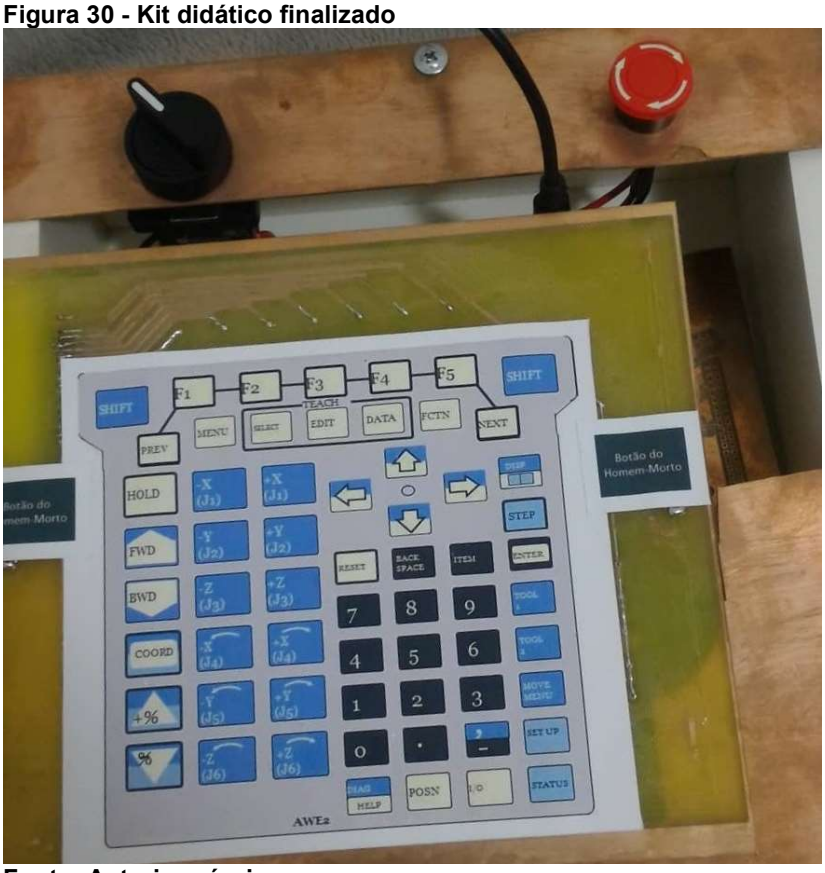
Figura 29 - Software 3DAutomate com o robô em posições diferentes



Fonte: Autoria própria

Na Figura 30, uma foto do kit didático já finalizado com o teclado, os botões de homem-morto, emergência e ON/OFF, assim como o cabo para comunicação serial que deve ser conectado no computador.

Figura 30 - Kit didático finalizado



Fonte: Autoria própria

6 CONCLUSÃO

O desenvolvimento de um kit didático microcontrolado para ensino de robótica foi motivado pela necessidade dos alunos treinarem os conhecimentos adquiridos em um curso de robótica e, pela constatação de que muitas faculdades não conseguem possibilitar esta prática tão importante aos estudantes. Para isso, o kit deve possibilitar a visão dos movimentos do braço em suas diferentes coordenadas, mostrar a função da junta para atingir o local final e apresentar a funcionalidade de um *Teach Pendant*.

A partir da execução deste projeto, foi possível conhecer melhor as opções de microcontroladores disponíveis com o intuito de escolher um componente que fosse adequado para este sistema embarcado. Na fase de testes, percebe-se que o microcontrolador funcionou conforme era esperado, lendo as entradas necessárias e determinando a saída via comunicação serial; o que confirma que a escolha foi apropriada.

Além disso, durante este trabalho, foi necessário adquirir conhecimento sobre robótica para possibilitar então, o desenvolvimento de um kit didático que respeite as leis robóticas e o método de funcionamento do robô Fanuc LR Mate 200iC. Desta maneira, as pessoas beneficiadas pelo uso do kit, irão conhecer o funcionamento do *Teach Pendant* sem perdas por não utilizar o verdadeiro sistema robótico.

Um problema encontrado foram os botões de contato, que não tiveram o funcionamento esperado. Entretanto, estes foram os únicos botões encontrados compatíveis com o teclado de membrana.

Foi possível constatar que o kit pode ser utilizado com a finalidade proposta em salas de aula, assim como ser adquirido por quem cursa uma faculdade a distância. A utilização do kit didático, desde a instalação até o teste de funcionamento do robô é simples e fácil de ser ensinada. Garantindo que qualquer pessoa, mesmo sem experiência, possa manusear sem dificuldades.

6.1 Sugestão de Trabalhos Futuros

Para tornar o kit ainda mais parecido com o *Teach Pendant* existe um periférico que pode ser adicionado, uma tela LCD que mostre: qual tecla está sendo pressionada; as condições de segurança naquele momento; a posição e velocidade do robô; além de um histórico sobre a utilização do kit.

Outra oportunidade de estudo são os botões de contato, visto que estes não tiveram o desempenho de acordo com as expectativas. Pode ser analisado a troca por tipos diferentes de botão, mas que mantenham a funcionalidade parecida com o *Teach Pendant.*

REFERÊNCIAS

ABDERRAHMANE, M. S.; DJURIC A. M.; CHEN, W.; CP, Yeh. **Study and Validation of Singularities for a Fanuc LR Mate 200iC Robot**. 2014 IEEE, International Conference on Electro/Information Technology, pag. 432-437.

ALIEXPRESS. **36pcs Cross Shape Dustproof Metal dome array switch 12mm snap dome single Rohs New OEM order welcome.** Disponível em: <https://pt.aliexpress.com/item/36pcs-Cross-Shape-Dustproof-Metal-dome-array-switch-12mm-snap-dome-single-Rohs-New-OEM-order/32642689243.html>. Acesso em: 04 abr. 2017, A.

ALIEXPRESS. **On/Off Two 2 Position Rotary Select Selector Switch 1 NO 10A 600V AC.** Disponível em: <https://pt.aliexpress.com/item/On-Off-Two-2-Position-Rotary-Select-Selector-Switch-1-NO-10A-600V-AC/32704946552.html>. Acesso em: 04 abr. 2017, B.

ALIEXPRESS. **THGS-Red Mushroom DC 30V 5A AC 250V 3A Emergency Stop Push Button Switch.** Disponível em: <https://pt.aliexpress.com/item/THGS-Red-Mushroom-DC-30V-5A-AC-250V-3A-Emergency-Stop-Push-Button-Switch/32751647865.html>. Acesso em: 04 abr. 2017, C.

BALMANT, Ocimara. **Polos de ensino superior a distância crescem 133% em um ano.** Disponível em: <https://www1.folha.uol.com.br/educacao/2018/07/polos-de-ensino-superior-a-distancia-crescem-133-em-um-ano.shtml>. Acesso em: 09 set. 2018.

CABRAL. Eduardo L. L. **Análise de Robôs – Capítulo 5.** Disponível em: <http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf>. Acesso em: 29 set. 2018.

CRAIG, John J. **Robótica.** 3. ed. São Paulo, SP: Pearson, 2012. 379 p. ISBN 9788581431284.

CONTROLE E INSTRUMENTAÇÃO. **Painéis, Fios e Cabos visão geral básica.** Disponível em: <http://www.controleinstrumentacao.com.br/arquivo/ed_225/cp.html>. Acesso em: 18 maio 2017.

DENAVIT J.; HARTENBERG R. S. **A Kinematic Notation for Lower-pair Mechanisms Based on Matrices**. Journal of Applied Mechanics, 77: 215-221, 1955.

DENSO ROBOT. **Handling the Teach Pendant.** Disponível em: <https://densorobotics.com/content/user_manuals/19/005215.html>. Acesso em: 05 maio 2017.

DJURIC, Ana M.; WALTER, Scott E.; FERRI, Giulia A. **Virtual Teach Pendent Interface for Android Cellphone (VTPAC).** 2017. SAE, Detroit, ago. 2017.

EUROBOTS. **Fanuc LR Mate 200iC.** Disponível em: <https://www.eurobots.com.br/fanuc-robots-lr-mate-200ic-p179-pt.html>. Acesso em: 13 set. 2018.

FANUC. **ROBOT Products.** Disponível em: <https://www.fanuc.com/product/robot.html>. Acesso em 23 set. 2018.

FILIPEFLOP. **Placa FTDI FT232RL Conversor USB Serial.** Disponível em: <https://www.filipeflop.com/produto/placa-ftdi-ft232rl-conversor-usb-serial/>. Acesso em: 10 ago. 2018.

ISO. **ISO 10218-1:2011.** Disponível em: <https://www.iso.org/standard/51330.html>. Acesso em: 12 set. 2018.

MICROCHIP. **PIC18F2420/2520/4420/4520 Data Sheet.** Disponível em: < http://ww1.microchip.com/downloads/en/devicedoc/39631e.pdf>. Acesso em: 25 nov. 2016.

MICROCHIP. **8-bit PIC® and AVR® MCUs.** Disponível em: <http://www.microchip.com/design-centers/8-bit>. Acesso em 03 abr. 2017, A.

MICROCHIP. **Corporate Overview.** Disponível em: <http://www.microchip.com/about-us/company-information/about>. Acesso em 03 abr. 2017, B.

MICROCHIP. **MPLab.** Disponível em: <http://www.microchip.com/mplab>. Acesso em: 05 abr. 2017, C.

MICROCHIP. **PICkit™ 3 In-Circuit Debugger.** Disponível em: <https://www.microchip.com/Developmenttools/ProductDetails/PG164130>. Acesso em 19 set. 2018.

MORENO, Ana Carolina; FAJARDO, Vanessa. **Número de matrículas no ensino superior cresce 81% em dez anos.** Disponível em: <http://g1.globo.com/educacao/noticia/2013/10/numero-de-matriculas-no-ensino-superior-cresce-81-em-dez-anos.html>. Acesso em: 09 set. 2018.

NATIONAL INSTRUMENTS. **Conceitos gerais de Comunicação Serial.** Disponível em: <http://digital.ni.com/public.nsf/allkb/32679C566F4B9700862576A20051FE8F>. Acesso em: 18 abr. 2017.

ODEYINKA S.; DJURIC A. M. **Fanuc Family Inverse Kinematics Modeling, Validation and Visualization**. SAE Technical Paper 2016-01-0335, 2016.

PAIOTTI, Renato. **Comunicação Serial Usando o Protocolo RS232.** Disponível em: <http://www.newtoncbraga.com.br/index.php/eletronica/52-artigos-diversos/12095-comunicacao-serial-usando-o-protocolo-rs232-tel213>. Acesso em: 18 maio 2017.

PINHEIRO, Tarcisio C. F.; TRINDADE, Max R. P.; PANTOJA, Breno R. **CINEMÁTICA INVERSA DE UM MANIPULADO ROBÓTICO DE QUATRO GRAUS DE LIBERDADE UTILIZANDO MÉTODO NUMÉRICO ITERATIVO DA JACOBIANA PSEUDO-INVERSA.** Disponível em: <http://www.sbai2013.ufc.br/pdfs/7605.pdf>. Acesso em: 29 set. 2018.

PROTEUS. **Proteus Design Suite Overview.** Disponível em: <https://www.labcenter.com/>. Acesso em: 4 maio 2017.

PYO, S.; HASSAN, S.; JAN, Y.; YOON, J. **Design of 6-DOF Manipulator Intuitive Teaching System by Using Smart Phone Orientation**. 2013 4th International Conference on Intelligent Systems, Modelling and Simulation, pag. 364-369.

ROBOCORE. **Comparação Entre Protocolos de Comunicação Serial**. Disponível em: <https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html>. Acesso em: 10 set. 2018.

ROSÁRIO, João Maurício. **Robótica Industrial I: Modelagem, Utilização e Programação.** São Paulo: Baraúna, 2010.

SANT ANNA, Jose Paulo. **Robôs: Automação reduz perdas e aumenta produtividade com mais segurança.** Disponível em: <https://www.plastico.com.br/robos-automacao-reduz-perdas-e-aumenta-produtividade-com-mais-seguranca/>. Acesso em: 10 set. 2018.

SANTOS, Guilherme. **Robôs Industriais podem aumentar a produtividade e reduzir custos na Indústria Brasileira.** Disponível em: <https://www.automacaoindustrial.info/robos-industriais-podem-aumentar-produtividade-e-reduzir-custos-na-industria-brasileira/>. Acesso em: 10 set. 2018.

SILVEIRA, Cristiano Bertulucci. **Desvendando a comunicação RS232.** Disponível em: <https://www.citisystems.com.br/rs232/>. Acesso em: 18 maio 2017.

SMASHING ROBOTICS. **Most Advanced Robotics Simulation Software Overview.** Disponível em: <https://www.smashingrobotics.com/most-advanced-and-used-robotics-simulation-software/>. Acesso em: 14 maio 2017.

SOUZA, Fábio. **Microchip - Microcontroladores PIC de 8 bits.** Disponível em: <https://www.embarcados.com.br/pic/>. Acesso em: 20 abr. 2017.

VISUAL COMPONENTS. **Visual Components 4.1.** Disponível em: < https://www.visualcomponents.com/products/visual-components-4-0/>. Acesso em: 23 set. 2018.

WALTER S. E. **A Picture is worth a Thousand Words. VISUAL COMPONENTS – the 3D Product Suite.** Disponível em: <http://donar.messe.de/exhibitor/hannovermesse/2016/C171973/produktfolder-vc-englisch-eng-295742.pdf>. Acesso em 07 nov. 2016.

WALTER S. E. **CAD versus 3D Simulation 3D Simulation versus 3D Simulation for Manufacturing from Visual Components.** Disponível em: <http://www.visualcomponents.com/>. Acesso em: 07 nov. 2016.

ZANCO, Wagner da Silva**. Microcontroladores PIC18 com linguagem C: uma abordagem prática e objetiva com base no PIC18F4520**. 1. ed. São Paulo: Érica, 2010. 446 p. ISBN 978-85-36502854.

ANEXO A – Código em C para a programação do microcontrolador

```
/* Trabalho de Conclusão de Curso
 * Desenvolvimento de um kit didático microcontrolado para ensino de robótica
 * Author: Giulia Ferri
 */

// PIC18F4520 Configuration Bit Settings
// 'C' source line config statements
// CONFIG1H
#pragma config OSC = INTIO67    // Oscillator Selection bits (Internal oscillator block, port function on RA6 and RA7)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L
#pragma config PWRT = OFF       // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled))
#pragma config BORV = 3         // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H
#pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = PORTC   // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF     // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
#pragma config LPT1OSC = OFF    // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)
```

```
#pragma config MCLRE = ON       // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

// CONFIG4L
#pragma config STVREN = ON      // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L
#pragma config CP0 = OFF        // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)
#pragma config CP1 = OFF        // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)
#pragma config CP2 = OFF        // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)
#pragma config CP3 = OFF        // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)
```

```
#pragma config WRT3 = OFF        // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF        // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)
#pragma config WRTB = OFF      // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF      // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF       // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR1 = OFF       // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR2 = OFF       // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR3 = OFF       // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

#include <xc.h>
#include "delays.h"

#define C3 PORTAbits.RA0
#define C4 PORTAbits.RA1
#define C5 PORTAbits.RA2
#define C6 PORTAbits.RA3
#define L0 PORTAbits.RA4
#define L1 PORTAbits.RA5
```

```
#define LEDRED PORTAbits.RA6
#define LEDGREEN PORTAbits.RA7

#define SEG PORTBbits.RB0 //BOTÃO DE EMERGÊNCIA e DEAD MAN
#define L2 PORTBbits.RB1
#define L3 PORTBbits.RB2
#define L4 PORTBbits.RB3
#define L5 PORTBbits.RB4
#define L6 PORTBbits.RB5
#define L7 PORTBbits.RB6
#define L8 PORTBbits.RB7

#define C2 PORTCbits.RC0
#define C1 PORTCbits.RC1
#define C0 PORTCbits.RC2

//leitura do teclado
char getKey(void){
   char k = 80;

   C0 = 0; //Ativa coluna 0
   C1 = 1; //Desativa coluna 1
   C2 = 1; //Desativa coluna 2
   C3 = 1; //Desativa coluna 3
   C4 = 1; //Desativa coluna 4
   C5 = 1; //Desativa coluna 5
   C6 = 1; //Desativa coluna 6
   Delay1KTCYx(10);
   if(!L0)       k = 16; //confere se L0 igual a zero //SHIFT
   else if(!L1) k = 23; //NEXT
   else if(!L2) k = 31; //DISP
   else if(!L3) k = 32; //STEP
   else if(!L4) k = 46; //TOOL1
   else if(!L5) k = 53; //TOOL2
```

```
else if(!L6) k = 54; //MOVE
else if(!L7) k = 55; //SET UP
else if(!L8) k = 56; //HELP

C0 = 1; //Desativa coluna 0
C1 = 0; //Ativa coluna 1
Delay1KTCYx(10);
if(!L0)       k = 15; //F5
else if(!L1) k = 22; //FCTN
else if(!L2) k = 29; //RIGHT
else if(!L3) k = 42; //ENTER
else if(!L4) k = 9;
else if(!L5) k = 6;
else if(!L6) k = 3;
else if(!L7) k = 58; //VIRG
else if(!L8) k = 57; //POSN

C1 = 1; //Desativa coluna 1
C2 = 0; //Ativa coluna 2
Delay1KTCYx(10);
if(!L0)       k = 14; //F4
else if(!L1) k = 21; //DATA
else if(!L2) k = 28; //UP
else if(!L3) k = 30; //DOWN
else if(!L4) k = 8;
else if(!L5) k = 5;
else if(!L6) k = 2;
else if(!L7) k = 61; //PONTO
else if(!L8) k = 59; //I/O

C2 = 1; //Desativa coluna 2
C3 = 0; //Ativa coluna 3
Delay1KTCYx(10);
if(!L0)       k = 13; //F3
```

```
else if(!L1) k = 20; //EDIT
else if(!L2) k = 27; //LEFT
else if(!L3) k = 40; //BACK
else if(!L4) k = 7;
else if(!L5) k = 4;
else if(!L6) k = 1;
else if(!L7) k = 0;
else if(!L8) k = 60; //STATUS

C3 = 1; //Desativa coluna 3
C4 = 0; //Ativa coluna 4
Delay1KTCYx(10);
if(!L0)       k = 12; //F2
else if(!L1) k = 19; //SELECT
else if(!L2) k = 26; //TX+
else if(!L3) k = 35; //TY+
else if(!L4) k = 38; //TZ+
else if(!L5) k = 45; //RX+
else if(!L6) k = 49; //RY+
else if(!L7) k = 52; //RZ-
else if(!L8) k = 39; //RESET

C4 = 1; //Desativa coluna 4
C5 = 0; //Ativa coluna 5
Delay1KTCYx(10);
if(!L0)       k = 11; //F1
else if(!L1) k = 18; //MENU
else if(!L2) k = 25; //TX-
else if(!L3) k = 34; //TY-
else if(!L4) k = 37; //TZ-
else if(!L5) k = 44; //RX-
else if(!L6) k = 48; //RY-
else if(!L7) k = 51; //RZ-
else if(!L8) k = 41; //ITEM
```

```
    C5 = 1; //Desativa coluna 5
    C6 = 0; //Ativa coluna 6
    Delay1KTCYx(10);
    if(!L0)       k = 10; //SHIFT1
    else if(!L1) k = 17; //PREV
    else if(!L2) k = 24; //HOLD
    else if(!L3) k = 33; //FWD
    else if(!L4) k = 36; //BWD
    else if(!L5) k = 43; //COOD
    else if(!L6) k = 47; //SPEED+
    else if(!L7) k = 50; //SPEED-

    Delay1KTCYx(10); //debounce
    return (k);
}

//rotina principal
void main(void) {
    // Configurações dos pinos de I/O
    TRISA = 0b00110000;  //saída os LEDs e as colunas
    TRISB = 0xFF; //entrada
    TRISC = 0xF8;  //RC6 e 7 configurados como entrada para a porta serial
    TRISD = 0xFF; //
    TRISE = 0xFF;
    ADCON1 = 0x0F; //todas as portas digitais
    PORTA = 0x00;
    PORTC = 0xFF;

    //config comunicação
    TXSTAbits.BRGH = 1; //
    TXSTAbits.SYNC = 0; //Comunicação assíncrona
    TXSTAbits.TXEN = 1; //Habilita Transmissão
    RCSTAbits.SPEN = 1; //Habilita Porta Serial
```

```
BAUDCONbits.BRG16 = 1; //BRG possui 16 bits
SPBRGH = 0;
SPBRG = 26;      //Bound Rate 9600

LEDGREEN = 1;  //LED pisca 2 vezes quando o kit é iniciado
LEDRED = 1;
Delay10KTCYx(10);
LEDGREEN = 0;
LEDRED = 0;
Delay10KTCYx(10);
LEDGREEN = 1;
LEDRED = 1;
Delay10KTCYx(10);

unsigned char x = 90;

//tratamento das informações recebidas
//envio via comunicação serial
while(1) {     //loop infinito
   LEDGREEN = 0;
   LEDRED = 0;
   x = getKey();

   if ((SEG == 1) && (x<80)) {  //dead man pressionado e botão de emergência solto
      LEDGREEN = 1;

      if (x == 0) {
         while(!TXSTAbits.TRMT);
         TXREG = '0';
      }
      else if (x == 1) {
         while(!TXSTAbits.TRMT);
         TXREG = '1';
      }
```

```
else if (x == 2) {
    while(!TXSTAbits.TRMT);
    TXREG = '2';
}
else if (x == 3) {
    while(!TXSTAbits.TRMT);
    TXREG = '3';
}
else if (x == 4) {
    while(!TXSTAbits.TRMT);
    TXREG = '4';
}
else if (x == 5) {
    while(!TXSTAbits.TRMT);
    TXREG = '5';
}
else if (x == 6) {
    while(!TXSTAbits.TRMT);
    TXREG = '6';
}
else if (x == 7) {
    while(!TXSTAbits.TRMT);
    TXREG = '7';
}
else if (x == 8) {
    while(!TXSTAbits.TRMT);
    TXREG = '8';
}
else if (x == 9) {
    while(!TXSTAbits.TRMT);
    TXREG = '9';
}
else if (x == 10) {
    while(!TXSTAbits.TRMT);
```

```
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'H';
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 11) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = '1';
}
else if (x == 12) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = '2';
}
else if (x == 13) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = '3';
}
else if (x == 14) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = '4';
}
```

```
else if (x == 15) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = '5';
}
else if (x == 16) {  //botão com problemas de contato
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'H';
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 17) {
    while(!TXSTAbits.TRMT);
    TXREG = 'P';
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'V';
}
else if (x == 18) {
    while(!TXSTAbits.TRMT);
    TXREG = 'M';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
```

```
    TXREG = 'N';
    while(!TXSTAbits.TRMT);
    TXREG = 'U';
}
else if (x == 19) {
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'C';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 20) {
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';

}
else if (x == 21) {
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
    while(!TXSTAbits.TRMT);
    TXREG = 'A';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'A';
}
else if (x == 22) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = 'C';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'N';
}
else if (x == 23) {
    while(!TXSTAbits.TRMT);
    TXREG = 'N';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'X';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 24) {
    while(!TXSTAbits.TRMT);
    TXREG = 'H';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
```

```
}
else if (x == 25) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '1';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 26) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '1';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 27) {
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 28) {
    while(!TXSTAbits.TRMT);
    TXREG = 'U';
    while(!TXSTAbits.TRMT);
    TXREG = 'P';
}
else if (x == 29) {
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'G';
    while(!TXSTAbits.TRMT);
    TXREG = 'H';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 30) {
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'W';
    while(!TXSTAbits.TRMT);
    TXREG = 'N';
}
else if (x == 31) {
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'P';
}
else if (x == 32) {
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'P';
}
else if (x == 33) {
    while(!TXSTAbits.TRMT);
    TXREG = 'F';
    while(!TXSTAbits.TRMT);
    TXREG = 'W';
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
}
else if (x == 34) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '2';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 35) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '2';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 36) {
    while(!TXSTAbits.TRMT);
    TXREG = 'B';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'W';
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
}
else if (x == 37) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '3';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 38) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '3';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 39) {
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
}
else if (x == 40) {
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'B';
    while(!TXSTAbits.TRMT);
    TXREG = 'A';
    while(!TXSTAbits.TRMT);
    TXREG = 'C';
    while(!TXSTAbits.TRMT);
    TXREG = 'K';
}
else if (x == 41) {
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'M';
}
else if (x == 42) {
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'N';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
}
else if (x == 43) {
    while(!TXSTAbits.TRMT);
    TXREG = 'C';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
    while(!TXSTAbits.TRMT);
    TXREG = 'D';
}
else if (x == 44) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '4';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 45) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '4';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 46) {
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
}
else if (x == 47) {
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 48) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '5';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 49) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '5';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 50) {
    while(!TXSTAbits.TRMT);
    TXREG = '-';
}
else if (x == 51) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '6';
    while(!TXSTAbits.TRMT);
    TXREG = '-';
```

```
}
else if (x == 52) {
    while(!TXSTAbits.TRMT);
    TXREG = 'J';
    while(!TXSTAbits.TRMT);
    TXREG = '6';
    while(!TXSTAbits.TRMT);
    TXREG = '+';
}
else if (x == 53) {
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'L';
    while(!TXSTAbits.TRMT);
    TXREG = '2';
}
else if (x == 54) {
    while(!TXSTAbits.TRMT);
    TXREG = 'M';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'V';
    while(!TXSTAbits.TRMT);
    TXREG = 'E';
}
else if (x == 55) {
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
```

```
        while(!TXSTAbits.TRMT);
        TXREG = 'E';
        while(!TXSTAbits.TRMT);
        TXREG = 'T';
        while(!TXSTAbits.TRMT);
        TXREG = 'U';
        while(!TXSTAbits.TRMT);
        TXREG = 'P';
    }
    else if (x == 56) {
        while(!TXSTAbits.TRMT);
        TXREG = 'H';
        while(!TXSTAbits.TRMT);
        TXREG = 'E';
        while(!TXSTAbits.TRMT);
        TXREG = 'L';
        while(!TXSTAbits.TRMT);
        TXREG = 'P';
    }
    else if (x == 57) {
        while(!TXSTAbits.TRMT);
        TXREG = 'P';
        while(!TXSTAbits.TRMT);
        TXREG = 'O';
        while(!TXSTAbits.TRMT);
        TXREG = 'S';
        while(!TXSTAbits.TRMT);
        TXREG = 'N';
    }
    else if (x == 58) {
        while(!TXSTAbits.TRMT);
        TXREG = 'V';
        while(!TXSTAbits.TRMT);
        TXREG = 'I';
```

```
    while(!TXSTAbits.TRMT);
    TXREG = 'R';
    while(!TXSTAbits.TRMT);
    TXREG = 'G';
}
else if (x == 59) {
    while(!TXSTAbits.TRMT);
    TXREG = 'I';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
}
else if (x == 60) {
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'A';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
    while(!TXSTAbits.TRMT);
    TXREG = 'U';
    while(!TXSTAbits.TRMT);
    TXREG = 'S';
}
else if (x == 61) {
    while(!TXSTAbits.TRMT);
    TXREG = 'P';
    while(!TXSTAbits.TRMT);
    TXREG = 'O';
    while(!TXSTAbits.TRMT);
    TXREG = 'N';
    while(!TXSTAbits.TRMT);
    TXREG = 'T';
```

```
            while(!TXSTAbits.TRMT);
            TXREG = 'O';
        }
        while(!TXSTAbits.TRMT);
        TXREG = 0x0D;
    }
    else {
        LEDRED = 1;
        while(!TXSTAbits.TRMT);
        TXREG = 'E';
        while(!TXSTAbits.TRMT);
        TXREG = 'R';
        while(!TXSTAbits.TRMT);
        TXREG = 'R';
        while(!TXSTAbits.TRMT);
        TXREG = 'O';

    }
  }
}
```

ANEXO B – Código em Python para a criação do servidor de comunicação

```
#Giulia Andrea Ferri
#Trabalho de conclusao de curso
#Desenvolvimento de um kit didatico microcontrolado para ensino de robotica

import socket
import sys
import serial
import time

def on_closing():
    top.destroy()
    if client:
        client.send('Exit')
        client.close()

#porta serial para conexao com micro
ser = serial.Serial('COM4',9600,timeout=0)

while 1:
    try:
        leitura = ser.readline()
        print leitura
        time.sleep(1)
    except ser.SerialTimeoutException:
        print('Kit nao conectado')
        time.sleep(1)

    #socket para conexao com 3DAutomate
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        port = 22000
        client.connect((socket.getfqdn(),port))
        client.send('Please Connect to me')
```

```
            client.settimeout(0.0)
            clientvc.send(leitura)

        except:
            client.close()
            client = None
```

ANEXO C – Código em Python para o robô no *3DAutomate*

```
from vcScript import *
import subprocess
import socket

#To make conection with python script
port = 22000

app = getApplication()
comp = getComponent()
robotIFace = comp.findBehaviour('RobotInterface')
robotComp = robotIFace.ConnectedComponent
robot = robotComp.findBehavioursByType( VC_ROBOTCONTROLLER )[0]

process = None
server = None

tpButton = comp.getProperty('Connect Teach Pendant')
if not tpButton:
  tpButton = comp.getProperty('Disconnect Teach Pendant')
  tpButton.Name = 'Connect Teach Pendant'
#endif

def teachPendant( button ):
  global server, process

  app.OnRender = None
  app.OnIdle = None

  if not server:
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Create a socket
object
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server.bind(('', port))     # Bind to the port '' means local host
```

```
    server.settimeout(0.0)
    process = subprocess.Popen([r'C:\Users\Giulia Ferri\tent2_tcc.py',str(comp.Port)],
shell=True)
    tpButton.Name = 'Disconnect Teach Pendant'
    print 'Teach pendant connected'
    app.startSimulation()
  else:
    print 'Teach pendant disconnected'
    server.close()
    server = None
    tpButton.Name = 'Connect Teach Pendant'

tpButton.OnChanged = teachPendant

def OnRun():
  global server,robotComp,robot

  if not server:
    return

  jogJoint = ['J1+', 'J1-', 'J2+', 'J2-', 'J3+', 'J3-', 'J4+', 'J4-', 'J5+', 'J5-', 'J6+', 'J6-']
  coord = 'Joint'
  step = 1.0

  while True:
    try:
      command, addr = server.recvfrom(1024)
      print command
      while True:
        try:
          command, addr = server.recvfrom(1024)

        except:
          break
```

```
except:
  delay(0.01)
  continue

if command == 'Coord':
    if coord == 'Joint':
      coord = 'World'
      print 'World Coords'
    elif coord == 'World':
      coord = 'Joint'
      print 'Joint Coords'
    #endif
elif command == '+':
    if step < 50:
      step = step + 1
    else:
      step = 50
    #endif
elif command == '-':
    if step == 1:
      step = 1
    else:
      step = step - 1;
    #endif
#endif


if (coord == 'Joint'):
  if command == 'J1 +':
    for i,j in enumerate(robot.Joints):
      robot.setJointTarget(i,j.CurrentValue)
    joint = 0
    sign = +1
    jprop = robotComp.getProperty(robot.Joints[joint].Name)
```

```
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J1 -':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 0
  sign = -1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()


elif command == 'J2 +':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 1
  sign = +1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J2 -':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 1
  sign = -1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()
```

```
elif command == 'J3 +':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 2
  sign = +1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J3 -':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 2
  sign = -1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()


elif command == 'J4 +':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 3
  sign = +1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J4 -':
```

```
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 3
  sign = -1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J5 +':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 4
  sign = +1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()

elif command == 'J5 -':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 4
  sign = -1
  jprop = robotComp.getProperty(robot.Joints[joint].Name)
  jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
  robot.setJointTarget(joint, jvalue )
  robot.move()


elif command == 'J6 +':
  for i,j in enumerate(robot.Joints):
    robot.setJointTarget(i,j.CurrentValue)
  joint = 5
```

```
    sign = +1
    jprop = robotComp.getProperty(robot.Joints[joint].Name)
    jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
    robot.setJointTarget(joint, jvalue )
    robot.move()

  elif command == 'J6 -':
    for i,j in enumerate(robot.Joints):
      robot.setJointTarget(i,j.CurrentValue)
    joint = 5
    sign = -1
    jprop = robotComp.getProperty(robot.Joints[joint].Name)
    jvalue = min(max(jprop.Value + sign*step,jprop.MinValue),jprop.MaxValue)
    robot.setJointTarget(joint, jvalue )
    robot.move()
  #endif

if (coord == 'World'):
  trg = robot.createTarget()
  m = trg.Target
  if 'J1+' in command:
     sign = +1
     m.translateAbs(sign*step,0,0)
  elif 'J1-' in command:
     sign = -1
     m.translateAbs(sign*step,0,0)
  elif 'J2+' in command:
     sign = +1
     m.translateAbs(0,sign*step,0)
  elif 'J2-' in command:
     sign = -1
     m.translateAbs(0,sign*step,0)
  elif 'J3+' in command:
     sign = +1
```

```
   m.translateAbs(0,0,sign*step)
elif 'J3-' in command:
   sign = -1
   m.translateAbs(0,0,sign*step)
else:
    p = m.P
    if 'J4+' in command:
      sign = +1
      m.rotateAbsX(sign*step)
    elif 'J4-' in command:
      sign = -1
      m.rotateAbsX(sign*step)
    elif 'J5+' in command:
      sign = +1
      m.rotateAbsY(sign*step)
    elif 'J5-' in command:
      sign = -1
      m.rotateAbsY(sign*step)
    elif 'J6+' in command:
      sign = +1
      m.rotateAbsZ(sign*step)
    elif 'J6-' in command:
      sign = -1
      m.rotateAbsZ(sign*step)
    #endif
    m.P = p
#endif
trg.Target = m
if trg.RobotConfig > -1:
    if trg.getConfigWarning(trg.RobotConfig) == VC_MOTIONTARGET_KW_OK:
      robot.moveImmediate(trg)
    #endif
#endif
```