Mecatrónica – 2013/2014

ROS - Detector de metais

Projecto

Luís Rocha 2010127532
José Medeiros 2010129934

Introdução

Neste trabalho foi adaptado o robô Roomba para a detecção de minas terrestres. Foi utilizado o ROS (Robot Operating System) para fazer o controlo deste robô. Com o ROS foi possível controlar os movimentos do robô através do teclado usando um terminal.

Para a detecção de minas foi utilizado o sensor *LCD1000EVM*. Este sensor devolve dois valores, o valor da proximidade e o valor da proximidade do metal. Quando na presença de metais o valor da proximidade aumenta. Com base neste princípio é possível detectar a presença de minas.

Por os manuais do sensor

Figura 1: Roomba e Sensor LDC100EVM

Todo list

Indíce

Indice de Figuras

1 Descrição do problema

Pretende-se detectar minas terrestres através do uso de um robô Roomba e de um sensor indutivo LCD1000EVM capaz de medir a proximidade e a frequência associada á presença de um metal nas imediações. Este sensor é composto por uma bobine em PCB, um LDC1000 que faz uso da bobine e um microcontrolador - MSP430.

Para ser possivel a concretização deste projecto é necessário implementar uma interface de comunicação entre o computador e o sensor para a aquisição de dados. Esta interface vai ser baseada num protocolo de comunicação que terá que ser o mesmo que é usado pelo microcontrolador. Será portanto necessário entender bem o protocolo visto que toda a base deste projecto se baseia neste sensor.

Por a referencia da imagem que mostra isso

Após a interligação entre o sensor e o computador através de uma porta USB é necessário combinar o robô Roomba com o sensor utilizando a plataforma ROS [1]. Para isso é obrigatório fazer uso de transformadas ($tf's$) para referenciar o sensor ao eixo/referencial em relação ao referencial principal do robô. Este passo é importantissimo para que se possa calcular as posições onde o sensor detecta minas.

Dizer que mais abaixo vou explicar o conceito das tf's

2 Conhecimentos Necessários

Após analisarmos os requisitos deste projecto chegámos á conclusão de que era necessário familiarizarmos com a plataforma ROS. Sendo assim seguimos os tutoriais existentes no website WikiRos [1]. Após concluidos os tutoriais, ficamos aptos a usar ROS usando C++ como linguagem de programação.

Os seguintes conceitos/utilidades foram usados no nosso projecto, a sua explicação breve apresenta-se abaixo:

Tf
: Estruturação dos referenciais do robô assim como as transformações necessárias entre os diferentes componentes [2] do robô para que se possa saber as posições de qualquer elemento num dado instante.

Marker
: Estrutura de dados que permite usar as bibliotecas do *rviz* para marcar minas no mapa. A posição das minas é obtida pela transformação da *tf* do sensor em relação ao referencial base do robô.

ArrayMarker
: Guardamos os markers (minas) que detectamos num array de markers. Para tal é necessário fazer *push_back* de um marker para o array quando é detectada a presença de uma mina.

Serial Port
: Protocolo de comunicação usado para comunicar com o sensor através de uma porta USB.

[1] Distribuição Hydro
[2] Sensores, plataformas, rodas, entre outros

3 Material disponível

Robô e API Roomba 555 com interface USB e *Wrapper* para comunicar com o robô

Sensor de Metal LDC100EVM

Computador Portatil de pequenas dimensões com o sistema operativo Ubuntu[3]

Bobine 1.5 mH

Condensador 19.2nF

ROS Plataforma ROS

[3]Ubuntu 13.04

4 LDC100EVM

O LDC100EVM é um sensor indutivo que permite medir a presença de metais retornando duas variavéis: posição do metal e a frequência de oscilação. É um circuito integrado que contém um microcontrolador MSP430 que é usado para fazer a interface entre o LDC (sensor) e um computador. Este integrado vêm com uma bobine que pode ser mudada se se partir a PCB na zona a tracejado e colocando ai uma bobine á nossa escolha. É possivel ver na figura 2 a zona a tracejado:

Figura 2: Zona a tracejado

Para dimensionar a bobine é necessário satisfazer duas condições para que o sensor possa funcionar na gama ideal:

1. A freqência de ressonância deve estar compreendida entre 5 kHz e 5 MHz
2. A resistência de ressonância[4] deve estar compreendida entre 798Ω e 3.93 MΩ

Para o calculo da freqência e a respectiva resistência usa-se as seguintes fórmulas:

$$f_{\text{ressonância}} = \frac{1}{2\pi\sqrt{LC}} \qquad (1) \qquad\qquad R_p = \frac{1}{R_s}\frac{L}{C} \qquad (2)$$

Tendo em conta (1) e (2) podemos dimensionar[3] uma nova bobine para o sensor. Para garantir que a bobine juntamente com o novo condensador estão em contacto com o sensor deve-se soldar um adaptador na placa que permite a fácil colocação de elementos através de uma chave de fendas.

Para calcular a indutância[2] medida pelo sensor é necessário fazer uso da frequência medida pelo LDC100EVM. Para tal:

$$L_{\text{medida}} = \frac{1}{C * (2\pi f_{\text{sensor}})^2} \qquad (3)$$

Codigo C++

Codigo Python

Codigo Matlab

[4]Simboliza as perdas associadas ás correntes de Eddy

5 Solução proposta

5.1 Hardware

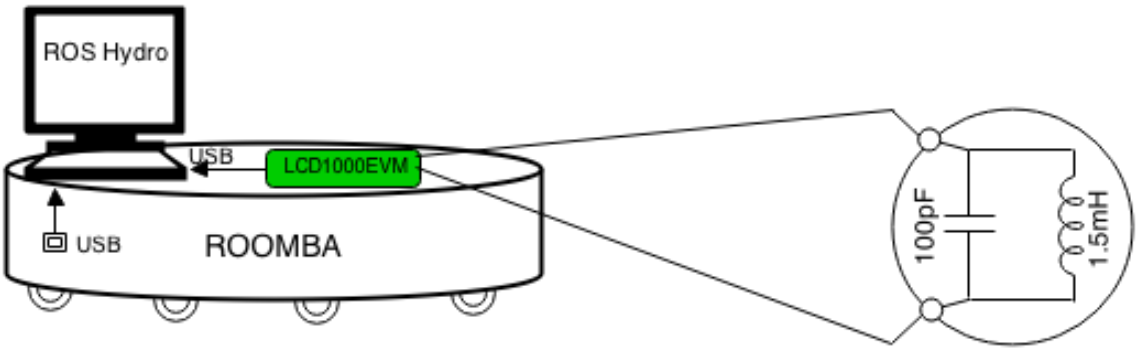

Figura 3: Estrutura

5.1.1 Arquitectura

Em termos de hardware para este trabalho foi utilizado um computador que vai controlar o Roomba e o sensor LCD1000EVM. No sensor foi ainda substituída a bobine que vinha com o mesmo por uma bobine e um condensador de valores por nós pretendidos.

5.1.2 Implementação

O robô Roomba encontra-se conectado por usb a um computador que o controla. Este computador encontra-se ainda ligado a um sensor LCD1000EVM que por sua vez tem ligado a si uma bobine e um condensador por nós projectados e adaptados.

Na presença de materiais conFdutores o valor de Rs e de L vão variar. Por intermédio da seguinte equação:

$$R_p = \frac{1}{R_s} \times \frac{L}{C} \tag{4}$$

é possível aproximar o circuito da seguinte forma:

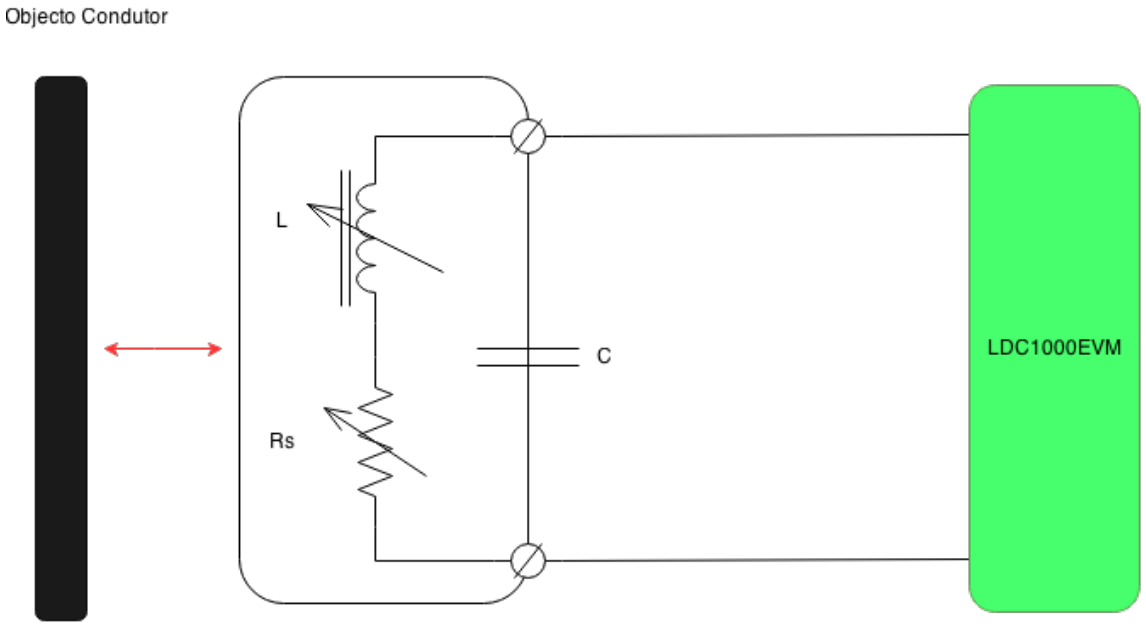

Figura 4: Circuito do Sensor

Existe apenas uma condição para o bom funcionamento deste circuito com este sensor para:

$$f_o = \frac{\omega_o}{2\pi} = \frac{1}{2\pi\sqrt{LC}} \tag{5}$$

f_o tem que estar compreendido entre 0.5Khz e 0.5Mhz.
Como tal desde que estas condições sejam cumpridas pode ser utilizada qualquer bobine, adaptando assim o sensor para diferentes situações por nós pretendidas. Tendo que ter em atenção que o sensor tem que se encontrar na gama de valores a cima referida, variando o valor do condensador garantimos que isso acontece.

É apenas necessário a alteração dos registos da placa, para que estejam adequados ao novo circuito. Os registos a serem modificados são relativos ao R_P, mais especificamente R_{PMAX} e R_{PMIN} e à frequência f_o. O método mais simples para

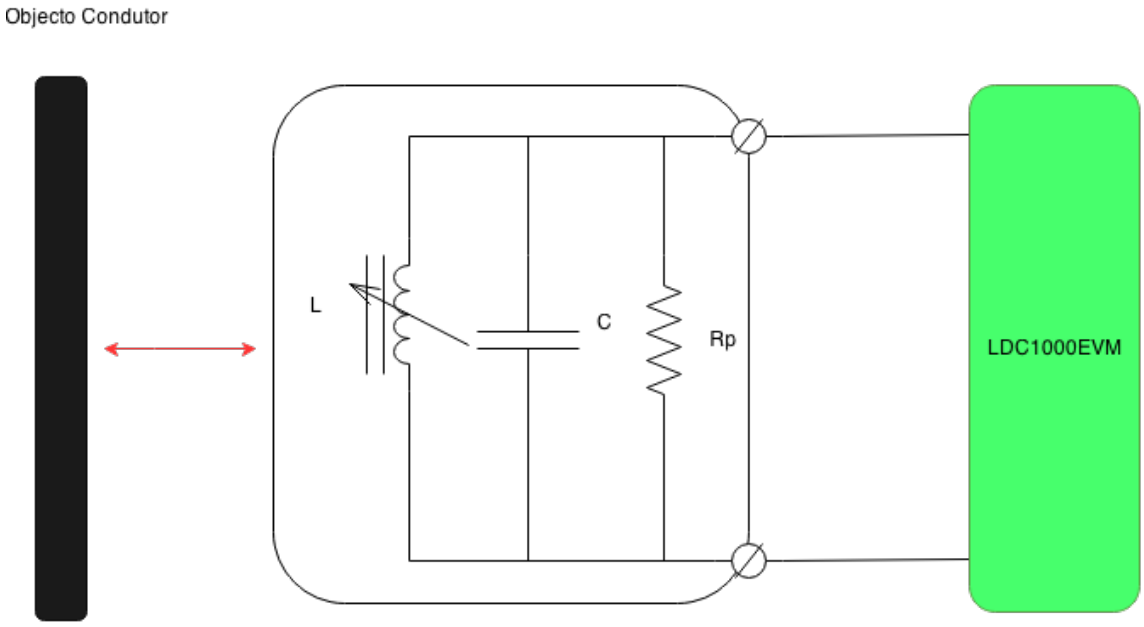

Figura 5: Circuito do Sensor

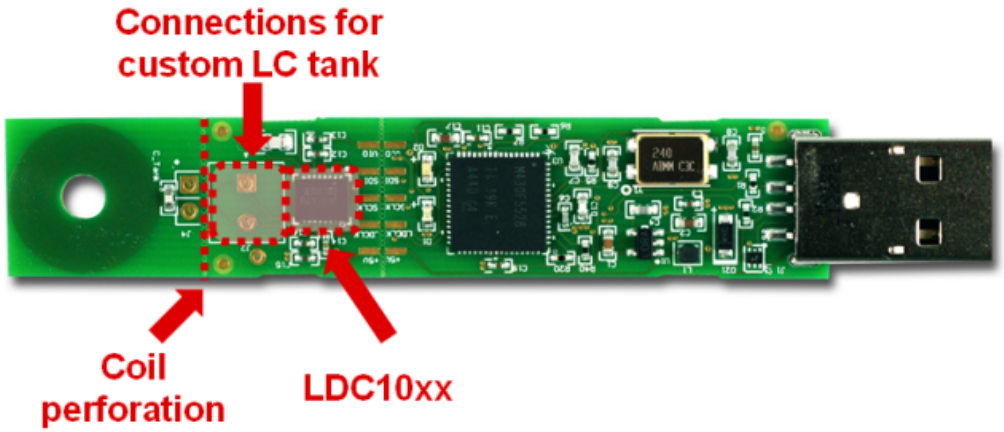

Figura 6: Circuito do Sensor

6 Solução proposta: Software -Implementação

6.1 Utilização do ROS

Utilizámos o Robot Operating System (ROS) como plataforma de comunicação entre os dois periféricos por nós utilizados e o computador. Com o ROS é possível implementar bibliotecas de cliente ROS como roscpp, rospy ou roslisp. Por outras palavras significa que facilmente se consegue adaptar programas criados em outras línguas (de entre C++, Python e LISP). No nosso caso em específico adaptámos o programa de leitura do sensor em cpp para a plataforma ROS. Numa segunda parte através do api disponibilizado pelo Gonçalo Cabrita e pelo Bruno Gouveia conseguimos controlar o Roomba pelo teclado do computador. Através da leitura dos encoders e do controlo dos motores foi possível controlar a velocidade angular e linear do robô.

Utilizando os nós do ROS foi possível assim controlar o robô e utilizá-lo para a deteção e marcação nas minas. De uma forma mais detalhada foi usada como referência a transformada "ODOM" que nos fornece a posição inicial em que o robô se encontra, com o nó entre esta transformada e o base link do Roomba sabe-se sempre a posição relativa à posição inicial. De seguida criando o nó da transformada sensor(criada por nós) é assim possível, com a informação devolvida pelo sensor, saber se o robô se encontra na presença de uma mina. Caso seja verificada a presença de uma mina são enviadas as coordenadas (x,y,z) para uma função "markers" que irá marcar no mapa as minas em relação a "ODOM", ou seja em relação à posição inicial do robô.

6.2 Comunicação com o sensor

A primeira parte deste trabalho foi concluir que o sensor utiliza porta série como protocolo de comunicação quando comunica com o computador. Como tal, para facilitar o uso futuro deste sensor criámos código para as leituras do sensor em três plataformas diferentes de programação. Adaptámos por isso o nosso código às plataformas *C++*, *Matlab* e *Python*. Estes três códigos encontram-se mais a baixo na tabela(Fazer referencia à tabela).

Analisando as partes mais importantes do código, o primeiro passo é definir a porta série pela qual estamos a comunicar, no nosso caso em particular foi a porta "'COM9'". De seguida definimos o tipo de controlo de fluxo, que neste caso é do tipo "software flow control". É necessário definir a frequência de comunicação, Baud Rate, que foi definida por nós a 9600. O próximo passo é definir a paridade e o número de bits da comunicação, neste caso não existe paridade e são utilizados 8bits na comunicação. Tal como na maioria dos dispositivos electrónicos na comunicação por porta série apenas é necessário de 1 "stop bit", foi ainda definido o tempo de espera para leitura máximo("Timeout") a 5 segundos. Definimos o modo de leitura contínua para funcionar de forma assíncrona. Por fim ao escrever na porta série o comando 0x33(hexadecimal) ou 3(ASCII), iniciamos a stream de dados.

7 Conclusões

Codigo C++

Referências

[1] Robot operating system. http://wiki.ros.org/, 2007. Hydro Edition.

[2] Texas Instruments. *Datasheet LDC1000*, 2013. http://www.ti.com/lit/ds/symlink/ldc1000.pdf.

[3] Texas Instruments. *Quick Start Guide*, 2013. http://www.ti.com/lit/ml/slyw022/slyw022.pdf.