Relatório T2 - Carlos Pakulski e Wagner Schettert

1. Introdução

O presente relatório possui como escopo apresentar como foi elaborado o trabalho II da disciplina de Programação de Periféricos, que consistia em desenvolver um sistema híbrido de Arduino e de Raspberry PI, o qual deveria ter as seguintes características:

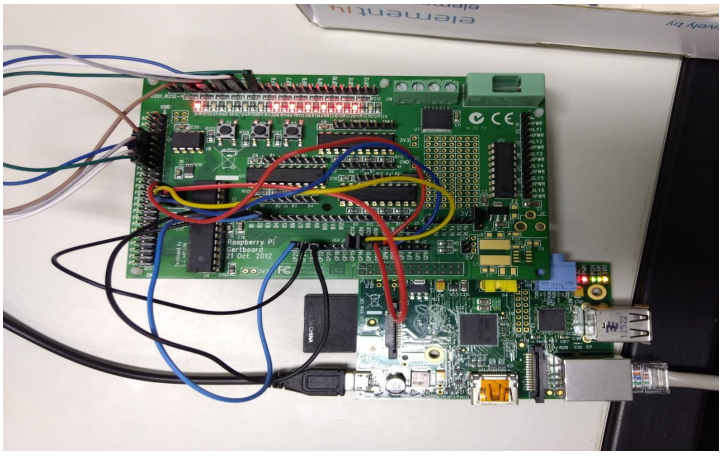
a) Reconhecer e escutar a função de apertar os 3 botões integrados à placa GertBoard em um software rodando no Raspberry Pi;
b) Dois dos botões seriam designados a mudar o padrão de acionamento de 5 LEDs também integrados a GertBoard;
c) O terceiro botão deveria apresentar na tela a quantidade de vezes em que cada padrão rodou no sistema.

Para tanto, restou convencionado que o software que rodava no Arduino era o responsável por acionar os padrões dos LEDs, bem como em fazer a contagem das vezes em que os mesmos rodavam. Não menos importante, o sistema do Arduino deveria mandar os contadores via serial para o software que rodava no Raspberry Pi. Em se tratando do sistema do Raspberry, esse seria o responsável por fazer a escuta e leitura dos eventos dos três botões mencionados anteriormente, mandando os eventos adiante via serial para o software do Arduino. O programa rodando no Raspberry deveria também aguardar os dados enviados de volta pela serial pelo sistema do Arduino (trazendo os contadores).

2. Configuração

A configuração feita pela dupla envolveu primeiramente instalar um Linux embarcado na Raspberry Pi que contivesse compilador C/C++, e que comportasse integração com Arduino. Após instalado o sistema Linux, foram feitas diversas configurações, tanto a nível de hardware quanto a nível de software, sendo elas:

A. Hardware
    a. Foram conectados os cabos que ligam os botões da GertBoard (b1, b2, b3 IN) às portas 24, 23 e 22 do GPIO;
    b. Foram conectados os cabos que ligam os 5 primeiros LEDs aos pinos PB do microcontrolador;
    c. Também foram conectados cabos responsáveis por enviar os dados de programação do Arduino.



B. Software
    a. As pastas de entrada e saída do GPIO;
    b. O arquivo de configuração do compilador de Arduino utilizando Avrdude;
    c. Instalação e utilização do programa Minicom.

3. Software Arduino

O código feito em C++ para o Arduino foi dividido em três principais partes, uma delas responsável por esperar o dado enviado pelo Raspberry Pi sobre qual o botão apertado na GertBoard, outra contendo os diferentes padrões de iluminação dos LEDs e a terceira que era responsável por fazer o link entre os LEDs e os pinos PBx.

4. Software Raspberry Pi

O componente de software também elaborado em C++ que roda no Raspberry Pi é um pouco mais robusto do que o do Arduino. Ele controla os pinos GPIO de entrada e saída, e trata toda a lógica de apertar os botões. Para isso, utilizou-se um modo de controle dos GPIO para se reconhecer quando houve uma troca de sinais, o que interpretou-se como um evento de click. Após tratar o evento de click, o Raspberry tratou de escrever na porta serial o resultado do processamento do mesmo, enviando os caracteres “a”, “b” e “c” para o Arduino. Os dois primeiros caracteres seriam responsáveis por avançar e voltar nos padrões, e o “c” em mostrar os contadores.

5. Dificuldades Encontradas

A principal dificuldade encontrada pelo grupo foi no entender o porquê os dados que estavam sendo passados pela serial (UART) não estavam corretos de forma alguma. Muitas tentativas a nível de código foram feitas no intuito de consertar esse problema, quando na verdade o que aconteceu é que a configuração correta do Baud Rate do Arduino havia sido feita apenas no compilador da placa, e não da máquina local, onde de fato o software havia sido compilado. Após ter sido feita essa configuração também na máquina local, os dois softwares (Arduino e Raspberry Pi) começaram a se comunicar mais adequadamente através da porta serial. Porém, ainda estávamos recebendo dados incoerentes do Arduino. Casos onde o contador de padrões deveria apresentar 1 ou 2, eram recebidos como 49 e 50. Após algumas pesquisas na internet, percebemos que o comando utilizado no Arduino estava enviando o valor em ASCII para o software no Pi. Com isso, foi feito um simples tratamento no valor recebido.