Trabalho Prático de Compiladores[1]

Prof. Thiago Borges de Oliveira

[1] Especificações do Trabalho da disciplina de Compiladores, 2016/1

Conforme previsto no plano de ensino da disciplina, $\frac{1}{3}$ da média refere-se a um trabalho que consiste em criar um projeto de compilador (analisador léxico, sintático e semântico) que será integrado ao *backend* do LLVM para gerar código executável para Arduino Uno R3. Adicionalmente, deverá ser implementado um programa na linguagem criada, o qual será transferido para o Arduino e movimentará um braço robótico (Figura 1).

Figura 1: Braço robótico.

O aluno (ou grupo) deve:

1. Especificar uma linguagem de programação, de alto ou baixo nível, para controlar o braço robótico. O braço deve ser usado para executar alguma tarefa, como por exemplo, pegar ou levantar objetos. Deve haver uma entrada no sistema. Os sensores disponíveis serão: um botão, dois fotoresistores e um microfone;
2. Implementar o analisador léxico e o analisador sintático da linguagem. Implementar tratamento de erro no analisador sintático;
3. Implementar o analisador semântico[2] e emissão de código usando classes em c++, as quais serão fornecidas e estão descritas a seguir. Essas classes fornecidas controlam as portas de entrada e as portas de saída do Arduino;
4. Criar um programa na linguagem para executar uma tarefa com o robô;
5. Fazer uma apresentação da linguagem e do robô executando a tarefa em sala de aula; e
6. Durante a apresentação, destacar as características da linguagem, as dificuldades encontradas no projeto e o aprendizado.

[2] Tabela de símbolos e árvore sintática.

Especificação das Classes de Emissão de Código

As seguintes classes podem ser instanciadas e anexadas na árvore sintática, para produzir o código do programa:

- `Int8(char n)`: Cria uma constante inteira de 8 bits, de valor *n*;
- `Int16(short n)`: Cria uma constante inteira de 16 bits, de valor *n*;
- `Int32(int n)`: Cria uma constante inteira de 32 bits, de valor *n*;
- `Float(float n)`: Cria uma constante float, de valor *n*;
- `String(const char *s)`: Cria uma constante do tipo string, com o valor *s*;

- `Variable(const char *n, Node *e)`: Cria um variável de nome *n*, com o valor da expressão *e*. O parâmetro *e* deve ser um nó da árvore previamente criado, com uma instância de constante ou expressão aritmética;
- `Load(const char *n)`: Carrega o valor da variável *n* da memória, previamente criada com `Variable(n);`
- `InPort(const char *p)`: Lê o valor da porta de entrada *p* do Arduino;
- `OutPort(const char *p, Node *e)`: Seta o valor da porta de saída *p* para o valor defino pela expressão ou constante *e*;
- `Delay(Node *mseg)`: Inclui uma espera de *mseg* milisegundos. *mseg* pode ser uma constante ou expressão aritimética;
- `Print(Node *e)`: Imprime o valor da expressão *e* no display. O nó *e* pode ser do tipo Int, Float ou String.
- `BinaryOp(Node *l, char op, Node *r)`: Realiza uma operação binária (+, -, *, /) com os operandos *l* e *r*. *l* e *r* podem ser constantes ou expressões aritméticas.
- `CmpOp(Node *l, yytokentype op, Node *r)`: Realiza a operação de comparação *op*, entre os operando *l* e *r*. O operador *op* deve ser definido na linguagem léxica conforme a Tabela 1. *r* e *l* são constantes ou expressões aritméticas.
- `If(Node *e, Node *then, Node *else)`: Cria uma condição no programa, que avalia a expressão booleana *e* e desvia para *then* ou *else* conforme o resultado. Este é o comando *if* da linguagem.
- `While(Node *e, Node *stmts)`: Cria um laço de repetição no programa, que executa os comandos em *stmts*, enquanto a expressão *e* é avaliada como verdadeira. A expressão *e* deve ser uma instância da classe *CmpOp*. *stmts* deve ser do tipo *Stmts*.
- `Stmts(Node *ss)`: Classe que representa um comando ou um bloco de comandos, ou seja, um conjunto de instâncias de qualquer uma das classes acima descritas. Deve ser usada em uma regra recursiva, semelhante a regra da Lista 1. A classe Stmts possui um método auxiliar chamado *append*, para acrescentar statements em um bloco. Veja exemplo de uso no código robcmp, arquivo rob.y.
- `Program().generate(Node *n)`: Classe e método final que gera o código fonte intermediário. Deve ser chamada no símbolo inicial da gramática. *n* deve ser do tipo *Stmts*.

Tabela 1: Tokens e significado dos operadores de comparação.

Token	Operador
EQ_OP	==
NE_OP	!=
GE_OP	>=
LE_OP	<=
GT_OP	>
LT_OP	<

Lista 1: Exemplo de produções para Stmt, com regra recursiva.

```
stmts
 : stmts stmt { $$->append($2); }
 | stmt { $$ = new Stmt($1); }
 ;
```

Documentação do Projeto de Hardware

Segue abaixo o esquemático (Figura 2) e o projeto em *protoboard* (Figura 3) do *hardware* que será disponibilizado. Você pode realizar alterações no projeto, mas considere implementar inicialmente um compilador para esta especificação.

O *hardware* possui as entradas `MIC1`, `S1`, `R7` e `R8`, ligadas no Arduino, conforme especificado na Tabela 2. Existem 12 saídas, especificadas na Tabela 3.

Tabela 2: Descrição das entradas do *hardware*.

Entrada	Descrição
A0	Nível de luz no fotoresistor R7
A1	Nível de luz no fotoresistor R8
A2	Sinal do Botão S1
A3	Sinal do Microfone MIC1

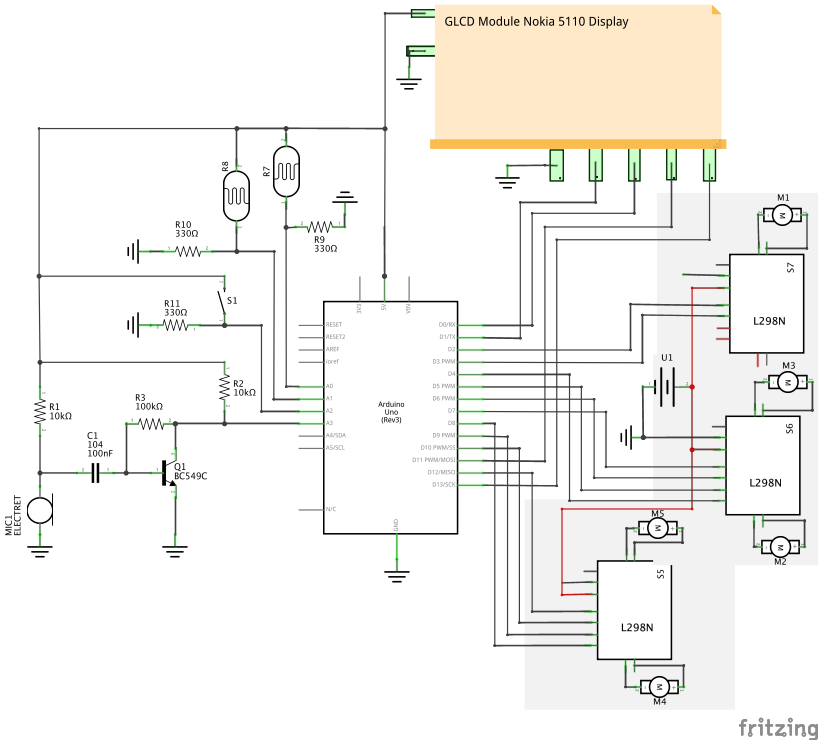

Figura 2: Esquemático do projeto de *hardware*.

Tabela 3: Descrição das saídas do *hardware*.

Saída	Descrição
D0	Display LCD Data/Command
D1	Display LCD Reset
D11	Display LCD MOSI
D13	Display LCD SCK
D2	Liga Motor 1 sentido horário
D3	Liga Motor 1 sentido anti-horário
D4	Liga Motor 2 sentido horário
D5	Liga Motor 2 sentido anti-horário
D6	Liga Motor 3 sentido horário
D7	Liga Motor 3 sentido anti-horário
D8	Liga Motor 4 sentido horário
D9	Liga Motor 4 sentido anti-horário
D10	Liga Motor 5 sentido horário
D12	Liga Motor 5 sentido anti-horário

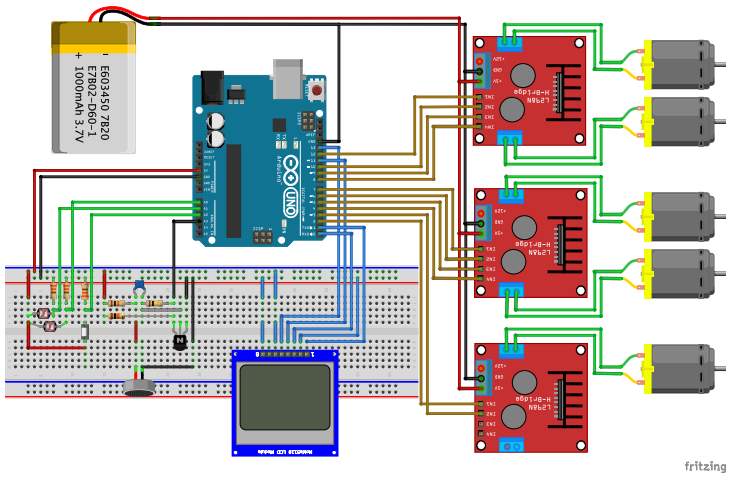

Figura 3: Projeto de *hardware* na *protoboard*.

Robcmp: Exemplo de Linguagem de Baixo Nível

O programa a seguir está escrito em uma linguagem de baixo nível, que possui comentários, atribuições, expressões aritméticas, condições simples (sem `and` e `or`) e laço de repetição. Esta linguagem pode ser usada para controlar o robô, porém é de propósito geral, ou seja, qualquer *hardware* ligado no Arduino poderia ser controlado com ela[3].

[3] O código completo desta linguagem e compilador está disponível em `http://github.com/thborges/robcmp`.

```
if (in0 > 10)      /* lê  porta 0 e verifica se é > 10*/
  out3 = 255;      /* seta porta 3 para 255 */

i = 1 / 1 + 0;     /* cria variável i com expressão aritimética */
while (i < 10) {   /* repete enquanto i < 10 */
    i = i + 1;     /* aumeneta o valor de i */
}

if (in1 - 10 > 10)  /* testa porta 1 */
  out6 = 255 - i;   /* seta porta 6 para 255 - i */
else
  out6 = 0;         /* seta porta 6 para 255 */
```