Marlboro College

MinNo:

A Language for Arduino Development Boards

By Logan H. G. Davis, Marlboro College Class of 2017

Abstract:

MinNo is a compiled language targeting the Arduino platform of development boards. It's an attempt to embed the distinction of the Arduino's "Modified Harvard" architecture principles at a more syntactic level than the Arduino Programming Language offers. The language's syntax emphasizes a more stylistically consistent form of programming that could be a better entry point for newcomers who are turned off by the stylistic variety of the Arduino Language/C++. MinNo's semantics are centered around more static allocation, limiting dynamic memory access and stateful idioms, avoiding many of the common points of failure that can make embedded system debugging so tedious. The language's compiler is written in Racket[1] and produces Arduino Language sketches for the ability to hand optimize performance critical sections in a more human-readable level than pure AVR assembly.

Implementation & Use :

The MinNo compiler is %100 Racket (formerly PLT-Scheme) source code.

The only dependencies for running the compiler from source are the following:

- Racket 6.8+
- BRAG (a popular LALR parser library for Racket).[2]

With those two dependancies and the MinNo compiler source code, you only need to issue the following command to compile .minno files:

```
$ racket <minno-source> <out-destination>
```

The Language:

MinNo takes syntactic notes from languages like Python and F#/ Scala to try an strike a balance between readability and clarity in specifying function and variable types.

The declaration of a mutable array of integers 1 through 5 in Arduino/C might look like:

```
int example_C_array[] = {1,2,3,4,5};
```

While in MinNo this would be written as:

```
let minno_array : mutable array[int] = [1,2,3,4,5];
```

Though the MinNo for mutable arrays in more verbose than C, mutability does not allow the use of one of the Arduino CPU's most district features: Program Memory. [3]

Program Memory is a separately addressed and accessed section of memory than the RAM and allows for quickly accessed & large storage of variables during run time.The catch is that program memory cannot be written to. MiNo's style and default behavior is meant to ease the process of using Program Memory.

Here is the same array example as above but the arrays will be placed in program memory and each example will access the first element after declaring it:

```
const int PROGREM example_C_array2[] = {1,2,3,4,5};
int example_access = pgm_read_word(&example_C_array[0]);
```

and again in MinNo:

```
let minno_array2 : array[int] = [1,2,3,4,5];
let example_access : int = minno_array2[0];
```

MinNo handling access to Program Memory allows for easier utilization of this feature of Arduino processors. The emphasis on immutable variables also helps to minimize state-related bugs that can be so hard to track down in an embedded environment.

Here is an example of a MinNo program that turns an LED on and off and the resulting C program when you compiler it:

Input:

```
#/ blink.minno
by Logan Davi
  An example blink program
/#
let outputPin : int = 13; //set pin for reference
def setup none -> none {
   pinMode outputPin OUTPUT; //set pin to OUTPUT mode
}
def loop none -> none {
   digitalWrite outputPin HIGH; //turn on
   delay 1000;
   digitalWrite outputPin LOW; //turn off
   delay 1000;
}
```

Output:

```
const PROGMEM int outputPin = 13 ;
void setup () {
    pinMode(pgm_read_word(&outputPin),OUTPUT) ;
 }
void loop () {
   digitalWrite(pgm_read_word(&outputPin),HIGH) ;
   delay(1000 ) ;
   digitalWrite(pgm_read_word(&outputPin),LOW) ;
   delay(1000 ) ;
 }
```

And here is the memory usage when compiling the output using the Arduino IDE:

```
Sketch uses 968 bytes (3%) of program storage space.
Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving
2039 bytes for local variables. Maximum is 2048 bytes.
```

For More Info:

Please feel free to visit the project's Github repo:

https://github.com/OkayAlright/ MinNo

There you will find documentation on the language, the compiler source code, and some reflective writing on making the language and compiler.

This compiler was made as part of Marlboro College's Undergraduate Plan of Concentration. Further information about the college and the Plan of Concentration can be found here:

https://www.marlboro.edu/academics/undergraduate/plan-of-concentration

References

[1]: "Racket." *racket-lang.org*. Accessed: April 1, 2017. https://racket-lang.org.

[2]: "brag: the Beautiful Racket AST Generator." *racket-lang.org*. Accessed: April 1, 2017. http://docs.racket-lang.org/brag/.

[3]: Camera, Dean. "GCC and the PROGMEM Attribute." *fourwalledcubicle.com* . Published: July 17, 2016. http://www.fourwalledcubicle.com/AVRArticles.php

Acknowledgements

I would like to thank Jim Mahoney for guidance and mentorship in implementing MinNo.

I would also like to thank Gavin Killough for his endless mockery of LISP which inspired me to implement the compiler in pure Racket.

Contact:
Email: ldavis@marlboro.edu
Twitter: @Death_by_kelp