Riptek HMTE 2022

MODUL PROGRAMMING ARDUINO

MINGGU 1

A. Tujuan

1. Menilik ulang dasar bahasa pemrograman C.
2. Memahami contoh kode yang diberikan.

B. Perlengkapan

1. Komputer.
2. Koneksi internet.

C. Rangkuman Teori dan Latarbelakang

1. Dengan bahasa apa Arduino diprogram?

Framework original Arduino menggunakan bahasa C dan C++ yang disederhanakan sebagai bahasa pemrogramannya. Bahasa tersebut digunakan karena cepat dan efisien untuk mikrokontroler. C/C++ adalah bahasa pemrograman tingkat tinggi, namun dekat dengan bahasa mesin.

Dengan komunitas yang semakin berkembang, Arduino tidak hanya dapat diprogram dengan C/C++. MicroPython membawa bahasa pemrograman terfavorit ke dunia mikrokontroler, termasuk Arduino. Pengembangan dalam bahasa Java, JavaScript, Assembly juga dapat dilakukan. Namun, bahasa C/C++ tetap menjadi bahasa terbaik untuk memulai belajar.

2. *Review* pemrograman Arduino

a. Arduino IDE

Untuk memprogram sebuah Arduino, diperlukan IDE (*Integrated Development Environment*). IDE ini berfungsi sebagai *manager library*/*module*, berkas-berkas kode, serta *compiler* bahasa C/C++ sebelum kode diunggah ke Arduino. IDE juga memberikan petunjuk jika ada kesalahan pada kode yang kalian tulis. IDE Arduino dapat diunduh melalui website resmi mereka (https://www.arduino.cc/en/software). Tersedia untuk komputer berbasis Windows, Linux, dan MacOS.

b. Struktur dasar

```
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Setiap kode Arduino memiliki dua fungsi yang wajib ada. Yaitu fungsi setup yang akan berjalan pertama kali dan hanya sekali, dan fungsi loop yang akan berjalan setelah fungsi setup dan akan berulang terus menerus.

Dalam bahasa C yang belum dimodifikasi, struktur tersebut ekuivalen dengan kode berikut:

```
int main(){

  setup();

  while (true){
    loop();
  };

}
```

Contoh kode di bawah ini dapat membantu kalian memahami perbedaan kedua fungsi tersebut.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // Inisiasi komunikasi serial dengan
                      // dengan kecepatan baud 9600
  println("Hello, world!");
  println("this is executed first and once");
  println("================================");
}
void loop() {
  // put your main code here, to run repeatedly:
  println("this is executed on and on");
  println("1"); println("2"); println("3");
  delay(700); // menghentikan eksekusi selama 700ms
}
```

c. Data dan variabel

Berikut ini adalah tipe data dalam bahasa C/C++ yang wajib kalian ketahui:

Type	**Keyword**	**Range**
Integer	int	-32768 – 32767
Boolean	bool, boolean	0/1 or true/false
Char	char	Alphabet, numbers, and symbols
Byte	byte	0 – 255
Float	float	-3.4028e+38 – 3.4028e+38
Void	void	none

Kumpulan elemen data dalam pemrograman disebut *array*. Dalam bahasa C/C++, setiap data dalam array harus memiliki tipe yang sama. Tiap elemen memiliki indeks yang dapat digunakan untuk mengakses elemen tersebut. Indeks array adalah bilangan bulat berurut yang dimulai dari 0.

Cara mendeklarasikan variable dan array pada bahasa C dalam Arduino IDE terdapat pada contoh di bawah. Kalian dapat mendeklarasikan variabel di luar maupun di dalam fungsi setup/loop.

```
bool x = true;
int  a  = -21;
char b = "A";
byte c = 20;
float d = 3.14;
float e = 2.37;
long int f = 1231;
unsigned int g = 31;

//deklarasi array
const int prime[] = {2, 3, 5, 7, 11};
float values[] = {2.8, .9, .002, 12.9};

//mengakses array
int num_one = prime[0];

void setup() {
  values[3] = 9.87; //mengubah nilai elemen array
}

void loop() {
}
```

d. Built-in function and library/module

Terdapat fungsi yang sudah tertanam dalam IDE Arduino dan tidak ada di bahasa C biasa. Fungsi-fungsi ini merupakan API (Application Programming Interface) untuk berinteraksi dengan fitur-fitur pada mikrokontroler, seperi pin, waktu, dan memori.

Tabel di bawah ini berisi beberapa contoh fungsi *built-in* yang akan sering kalian gunakan. Fungsi/API lainnya dapat kalian pelajari pada laman resmi Arduino (https://arduino.cc/en/Reference), **lampiran di halaman terakhir file ini juga berisikan beberapa fungsi tersebut**.

API	Args	Kegunaan
pinMode(pin, mode)	pin= nomor pin; mode= keyword mode pin, antara OUTPUT, INPUT, atau INPUT_PULLUP	Mengatur pin tertentu untuk berperilaku seperti input/output.
digitalRead(pin)	pin= nomor pin	Membaca nilai secara biner dari pin input.
digitalWrite(pin, value)	pin= nomor pin; value= nilai pin dapat berupa 0/1, HIGH/LOW, true/false.	Mengatur nilai dari pin output secara biner.
analogRead(pin)	pin= nomor pin	Membaca nilai secara analog dari pin input.
analogWrite(pin, value)	pin= nomor pin; value= nilai pin dalam jangkauan tertentu,	Mengatur nilai dari pin output secara analog.

	seperti 0-255 (bergantung pada resolusi sinyal analog)	

Modul built-in di Arduino yang akan sering kalian temui adalah sebagai berikut:

Module	**Kegunaan**
Serial	Mengatur komunikasi serial dengan protocol UART. Dapat digunakan saat Arduino berkomunikasi dengan komputer atau mikrokontroler lain. Komunikasi pada modul ini terbatas pada pin TX dan RX.
SoftwareSerial	Mengubah pin lain menjadi TX/RX untuk komunikasi UART. Digunakan Ketika pin TX/RX default tidak cukup untuk menangani komunikasi yang diperlukan.
Wire	Digunakan untuk protokol serial I2C. Protokol ini dapat menghubungkan beberapa perangkat dalam satu jalur komunikasi (pin SDA/SCL). Sering kali digunakan untuk membaca banyak sensor dan mengontrol display.
Servo	Digunakan untuk mengontrol perilaku motor servo dengan mudah.

D. Prosedur latihan

Di akhir modul in, ada beberapa program yang harus kalian coba sebagai latihan. Kode dapat diakses melalui repositori berikut: pelatihan_riptek2022. Terdapat folder dengan nama Part 1 dan Part 2. Folder tersebut yang akan kalian kerjakan.

PART 1:

1. Pada Part 1, gunakan IDE C online untuk memudahkan kalian mengeksekusi program. IDE dapat diakses pada laman: https://www.onlinegdb.com/online_c_compiler. Laman web akan langsung menampilkan workspace kalian. Kegiatan koding dapat dilakukan di file main.c. Pada bagian atas terdapat action bar, gunakan Run untuk menjalankan kode, dan stop untuk menghentikan eksekusi.
2. Copy dan paste kode dengan nama "1_deklarasi_var.c" ke dalam main.c pada IDE online. Kerjakan sesuai petunjuk yang terdapat pada comment, lalu jalankan.
3. Ulang untuk file berikutnya secara berurut.
4. Rekam/capture hasil program kalian.

PART 2:

1. Pada part ini, akan dilakukan simulasi Arduino secara online di website https://wokwi.com/ . Namun, kalian boleh mengimplementasikannya pada Arduino fisik, sehingga tidak perlu melakukan simulasi lagi.

2. Pada website, pilih Arduino Uno pada Start a New Project. Workspace kalian akan segera tampil.
3. Kosongkan file diagram.json pada workspace Wokwi, salin isi file diagram.json pada repositori ke diagram.json pada workspace tersebut.
4. Buka file dengan nama “1_blink_example.ino”, salin isinya ke sketch.ino pada workspace.
5. Lengkapi kode sesuai instruksi (jika ada), kemudian jalankan simulasi dengan menekan tombol play pada workspace.
6. Rekam/capture hasil simulasi.

Susun file-file hasil eksekusi program dan simulasi dengan rapih dan jelas. Zip (kompres) ke dalam satu file .zip/.rar. Kemudian kirim kan ke email riptekhmteunpad@gmail.com.

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
https://arduino.cc/en/Reference/

Structure & Flow

Basic Program Structure
```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures
```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions
```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Variables, Arrays, and Data

Data Types

bool	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	return type: no return value

Strings
```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays
```
byte myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```

Built-in Functions

Pin Input/Output
```
Digital I/O - pins 0-13 A0-A5
  pinMode(pin,
    {INPUT|OUTPUT|INPUT_PULLUP})
  int digitalRead(pin)
  digitalWrite(pin, {HIGH|LOW})

Analog In - pins A0-A5
  int analogRead(pin)
  analogReference(
    {DEFAULT|INTERNAL|EXTERNAL})

PWM Out - pins 3 5 6 9 10 11
  analogWrite(pin, value) // 0-255
```

Advanced I/O
```
tone(pin, freq_Hz, [duration_msec])
noTone(pin)
shiftOut(dataPin, clockPin,
  {MSBFIRST|LSBFIRST}, value)
shiftIn(dataPin, clockPin,
  {MSBFIRST|LSBFIRST})
unsigned long pulseIn(pin,
  {HIGH|LOW}, [timeout_usec])
```

Time
```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

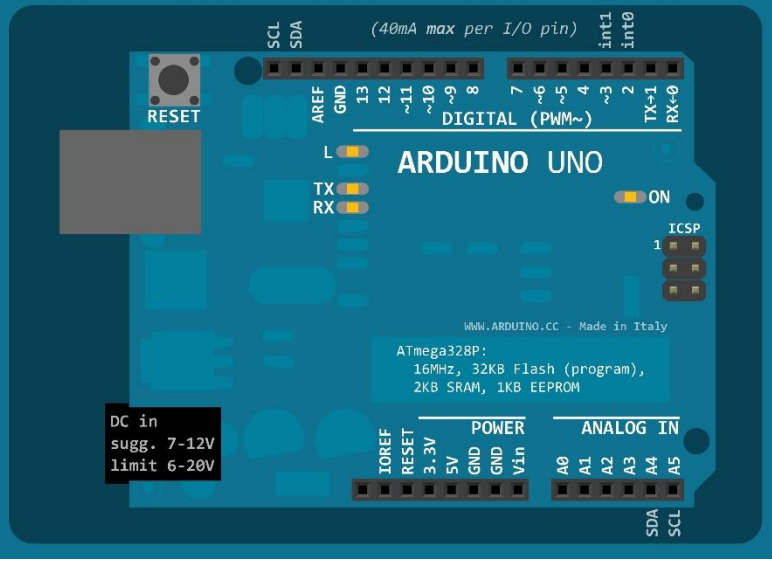
Math
```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers
```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions
```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts
```
attachInterrupt(interrupt, func,
  {LOW|CHANGE|RISING|FALLING})
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```



Libraries

Serial - *comm. with PC or via RX/TX*
```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - *comm. on any pin*
```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - *access non-volatile memory*
```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - *control servo motors*
```
attach(pin, [min_usec, max_usec])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - *I²C communication*
```
begin()      // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```


by Mark Liffiton
version: 2021-10-23

source: https://github.com/liffiton/Arduino-Cheat-Sheet/

Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

(end of file)