KULLIYYAH OF ENGINEERING

END-OF-SEMESTER EXAMINATION
SEMESTER I, 2023/2024 SESSION

Programme	:	**Mechatronics Engineering**	Level of Study	:	**UG 2**
Time	:	**2.30 pm - 5.30 pm**	Date	:	**1st February 2023**
Duration	:	**3 hours**			
Course Code	:	**MCTA 3203**	Section(s)	:	**1**
Course Title	:	**Mechatronics System Integration**			

This Question Paper Consists of **xx (Thirteen)** Printed Pages (Including Cover Page and Appendices) with **Part A: Twenty (20)** Multiple Choice Questions and **Part B: Three (3)** Questions.

.

INSTRUCTION(S) TO CANDIDATES

DO NOT OPEN UNTIL YOU ARE ASKED TO DO SO

- Total mark of this examination is **100.**
- This examination is worth **40 %** of the total course assessment.
- This question paper consists of two parts i.e. **Part A** and **Part B**. Answer **ALL QUESTIONS** in both parts.
- Only approved calculator with '**KoE approved**' sticker is allowed (non-programmable and non-graphical).
- Marks assigned to each question are listed in the margin.
- **Note that one of the conditions to pass the course is to obtain at least 50% of this examination.**

Any form of cheating or attempt to cheat is a serious offence which may lead to dismissal.

All electronics gadgets are prohibited in the exam hall / venue.
(e.g. mobile / smart phones, smart watches, and smart glasses)

PART A: Answer ALL questions (40 marks)

Choose the correct answer.

1. What does IDE stands for **(1)**

   A. In Deep Environment

   B. Integrated Development Environment

   C. Internal Deep Escape

   D. Integrated Development Enrolment

2. Which one of the following is the name of the recommended operating system used for Raspberry Pi? **(1)**

   A. Ubuntu

   B. NOOBS

   C. Debian

   D. Raspbian

3. Raspberry Pi board are _____. Arduino board are_____, **(2)**

   A. Microcontroller, Microprocessor

   B. Microcontroller, Microcontroller

   C. Microprocessor, Microcontroller

   D. Microprocessor, Microprocessor

4. In writing Arduino sketch, forward slash stands for **(1)**

   A. code

   B. comments

   C. input

   D. output

5. Which chip is used in Arduino UNO? **(1)**

   A. ATmega2560

   B. AT91SAM3x8E

   C. C. ATmega32114

   D. ATmega328p

6. If a DC motor is rotating clockwise with motor_pin1=LOW and motor_pin2=HIGH, how can we reverse its direction? **(2)**

   A. motor_pin1=LOW and motor_pin2=HIGH
   B. motor_pin1=HIGH and motor_pin2=LOW
   C. motor_pin1=HIGH and motor_pin2=HIGH
   D. motor_pin1=LOW and motor_pin2=LOW

7. In Arduino, a function is a series of programming statements that can be called by name. Which command is called once when the program starts? **(2)**

   A. loop( )
   B. (output)
   C. setup( )
   D. (input)

8. **int LED1=4;**
   what is this statement? **(2)**

   A. Voidloop
   B. Voidsetup
   C. Function
   D. Declaration

9. Which command is called repetitively over and over again as long as the Arduino has power. **(2)**

   A. (output)
   B. loop ( )
   C. setup ( )
   D. (input)

(3)

10. The following code is written:

```
Serial.print ("Hello");
Serial.println ("world!");
Serial.print( ":D");
```

what would appear on the serial monitor after running the code?

A. Hello world! :D
B. Helloworld!
   :D
C. Hello
   world! :D
D. Hello
   World!
   :D

(2)

11. What does delay(10000) result in?

A. 10 minutes
B. 100 seconds
C. 10 seconds
D. 10000 seconds

(2)

12. What function is used to provide a value between an 8-bit resolution used for dimming lights or setting speed of a motor?

A. analogRead
B. Setup.write
C. digitalWrite
D. analogWrite

(2)

13. Name data type used for numbers with lots of decimals?

A. string
B. int
C. float
D. bool

14. What is the significance of SD card when inserted into Rasberry Pi? **(2)**

A. It serves as a processor

B. It serves as a RAM

C. It serves as Hard Drive

D. None of the above

15. What function used to read the state of a digital pin? **(2)**

A. digitalWrite

B. pinMode

C. digitalRead

D. serial.Read

16. What will be the output for the phyton program below? **(3)**

```
height= float(2.5)
length = float(8.45)
width = float(6.5)
tot = height*length*width
print("Cuboid Volume", tot)
```

2.5 x 8.45 x 6.5 =

A. 137.313

B. Cuboid 54.93

C. Cuboid Volume 137.313

D. Cuboid Volume is 137.313

17. Which of the following which is not a fundamental component of IoT? **(2)**

A. Sensors

B. Transformers

C. User interface

D. Connectivity and data processing

(2)

18. Identify the incorrect advantage of IoT.

A. Cost efficient

B. Enhanced data collection

C. Privacy and security

D. Improve customer engagement

19. For phyton code below, choose the correct output. **(3)**

```
Student_Names= ["Johan", "Zaki", "Chai", "Adi", "Rina", "Tini"]
Student_Names.sort()
player.say(Student_Names)
```

A. Adi, Chai, Johan, Rina, Tini, Zaki

B. Adi, Johan, Chai, Rina, Tini, Zaki

C. Zaki, Tini, Rina, Johan, Chai, Adi

D. Tini, Zaki, Chai Rina, Johan, Adi

20. For the following code, identify which row produces error **(3)**

```
int Led1=4;                    //row 1
void setup(){                  //row 2
digitalWrite (LED1, High); //row 3
}                              // row 4
```

A. row 2

B. row 3

C. row 4

D. row 1

PART B: Answer ALL questions (60 marks)

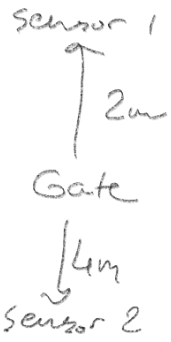
1. Design a system that is able to control three LEDs in such a way that LED1 blinks every 0.5s, LED2 blinks every 1s, and LED3 blinks every 3s. Demonstrate this task by using Arduino Uno, and other relevant components. You are required to

   i. Identify all required components. **(3)**

   ii. Draw neatly showing the complete connections between the arduino, and all other relevant components (**please use figures in Appendix 1**) **(7)**

   iii. Write complete codes including comments for each lines. **(10)**

2. A parking gate system for cars comprising two sensors and an actuator, with seamless integration through an Arduino microcontroller for efficient data processing and actuator control must be developed. Sensor 1 is strategically positioned 2 meters in front of the gate, while sensor 2 is placed 4 meters from the gate's backside. The operational logic is structured as follows: when sensor 1 detects an approaching car, the actuator (gate opener) is triggered, initiating gate opening. Conversely, if sensor 2 registers a departing car from the gate, the actuator is deactivated, leading to gate closure.



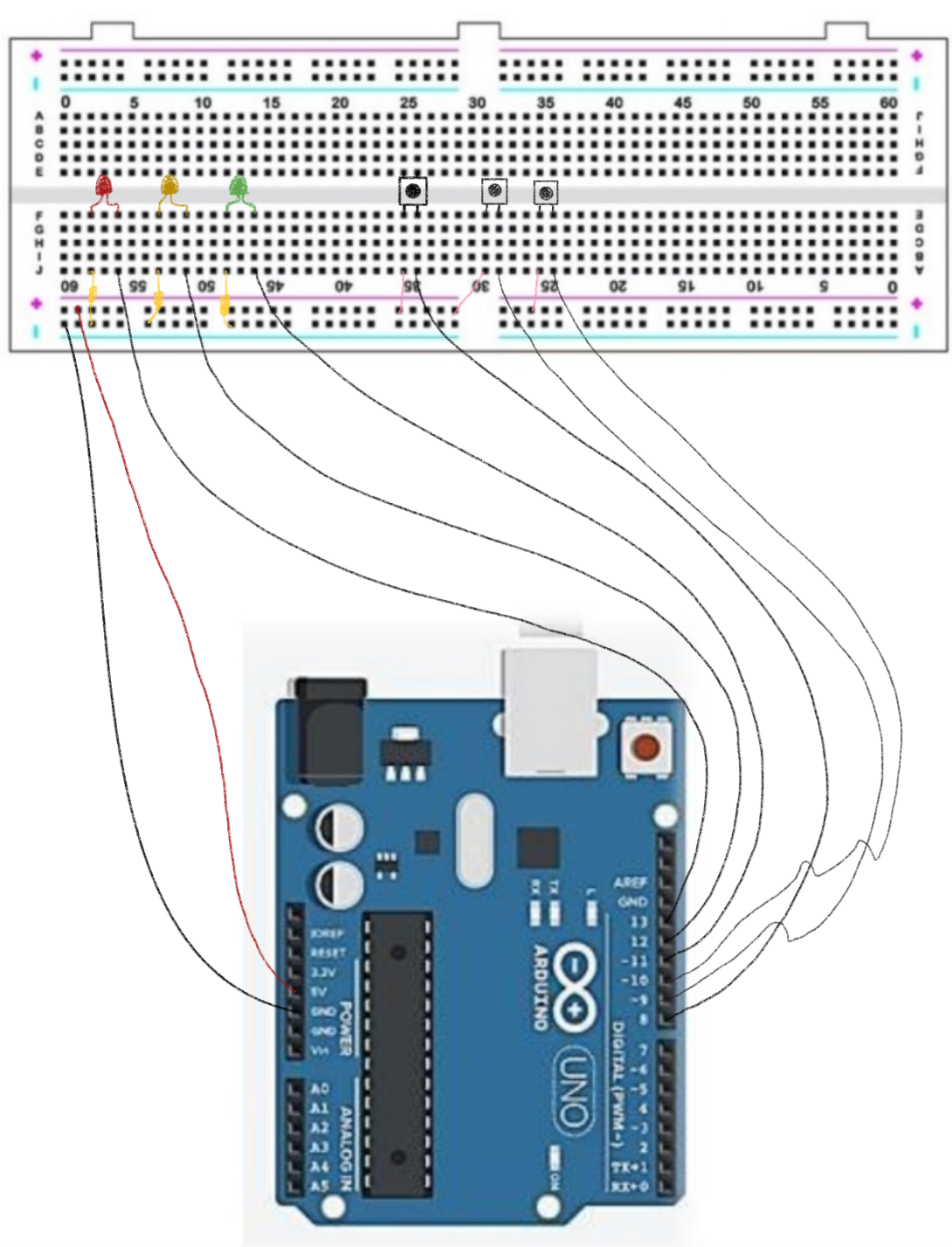
   i. Create an Arduino sketch for the system described above. Complete your code with comments. **(10)**

   ii. An easy-to-use interface has to be implemented to monitor the status of the sensors and actuator, ensuring effortless operation and real-time feedback for users Develop a simple User Interface for the operation of the system using Python. Complete your code with comments. **(10)**

1.

i. - Arduino Uno
- RED LED
- Green LED
- Yellow LED
- 3 Resistors 220Ω
- 1 Push bottun
- jump wires
- bread board

ii.




```
int const ledR = 13;
int const ledY = 12;
int const ledG = 11;
int const Bottun1 = 8;
int const Bottun2 = 9;
int const Bottun3 = 10;
Void setup() {
pinMode(ledR, OUTPUT);
pinMode(ledY, OUTPUT);
pinMode(ledG, OUTPUT);
pinMode(Bottun1, INPUT);
pinMode(Bottun2, INPUT);
pinMode(Bottun3, INPUT);
}
void loop() {
 if (digitalRead(Bottun1) == HIGH){
digitalWrite(ledR, HIGH);
delay(500);
digitalWrite(ledR, LOW);
delay(500);
} else if (digitalRead(Bottun2) == HIGH){
digitalWrite(ledY, HIGH);
delay(1000);
digitalWrite(ledY, LOW);
delay(1000);
} else if (digitalRead(Bottun3) == HIGH){
digitalWrite(ledG, HIGH);
delay(3000);
digitalWrite(ledG, LOW);
delay(3000);
}}
```

```
#include <Servo.h>
const int irPin= 7;
const int irPin2= 8;

Servo gateservo;

void setup(){
 Serial.begin(9600);
 pinMode(irPin,INPUT);
 pinMode(irPin2, INPUT);
 gateservo.attach(6);
 gateservo.write(0);
}

void loop(){
 int irstate=digitalRead(irPin);
 int irstate2=digitaRead(irPin2);
 if(irstate==LOW){
  Serial.println("Gate is opening for approaching car");
  gateservo.write(90);
 }
if (irstate2==LOW){
  Serial.println("Gate is closing for departing car");
  gateservo(0);
 }

 if(Serial.available()>0){
  String cmd=Serial.readStringUntil('\r');
  if (cmd=="OPEN"){
   gateservo(90);
  }
  if (cmd=="CLOSE"){
   gateservo(0);
  }
 }
 delay(100);
}
-------------------------------------------------------------------------------------------
import time
import serial
ser=serial.Serial('COM7',9600)
time.sleep(1)

try:
   while True:
     while(ser.inWaiting()==0):
       pass
      data=ser.readline()
      data=str(data, 'utf-8')
      data=data.strip('\r\n')
      print(data)
      user_input=input('Enter "OPEN" to open the gate, and "CLOSE" to close it ')
      user_input=user_input+'\r'
      ser.write(user_input,encode())

except KeyboardInterrupt:
   print('Exiting program')
ser.close()
```

3. The university's student union is hosting an annual meeting for its members and needs an authentication system for venue access. This system allows entry only for those with valid member cards and includes a reminder for yearly membership renewals. Members who renew successfully get a free drink from a vending machine linked to the authentication system. The drink dispenser is operated by a single actuator. To align with the specified requirements, a system that integrates an RFID card reader, a vending machine, and utilizes Arduino boards and Python for implementation must be designed. Python code will be used to improve the user friendliness of the system's operation.

   i. Write an Arduino sketch to meet the above requirements. Complete your code with necessary comments. **(10)**

   ii. Write the necessary Python code. Complete your code with comments. **(10)**

-END OF PAPER -

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

#define ssPin 10
#define rstPin 9
MFRC522 mfrc(ssPin, rstPin);  // Create MFRC522 instance.
Servo drinkservo;

void setup(){
 Serial.begin(9600);
 SPI.begin();  // Initiate SPI bus
 mfrc.PCD_Init();  // Initiate MFRC522
 drinkservo.attach(6);
 drinkservo.write(0);
 Serial.println("System initialized");
}

void loop(){
 if(!mfrc.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial()){
  delay(50);
  return;
 }
 Serial.print("UID tag: ");
 String content= "";
 byte letter;
 for (byte i = 0; i<mfrc.uid.size; i++) {
  Serial.print(mfrc.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc.uid.uidByte[i], HEX);
  content.concat(String(mfrc.uid.uidByte[i] < 0x10 ? " 0" : " "));
  content.concat(String(mfrc.uid.uidByte[i], HEX));
  }
 content.toUpperCase();  //covert ID to uppercase
 if (content.substring(1) == "BD 31 15 2B") {
   Serial.println("Authorized access");
   if(Serial.available()>0){
    String cmd=Serial.readStringUntil('\r');
    if(cmd==="Okay"){
     Serial.println("Thank you for the renewal");
     drinkservo.write(90);
     Serial.prinln("Drink dispensed");
    if(cmd=="NO"){
     Serial.println("You are free to enter"); }
   } }
 else {
  Serial.println("Access denied");
 }
}
```

```
import serial
ser=serial.Serial('COM17',9600)
try:
  while True:
       while(ser.inWaiting()==0):
          pass
        data=ser.readline()
       data=str(data, 'utf-8')
       data=data.strip('\r\n')
        print(data)
       mycmd=input('Please renew the membership: (YES,NO)')
   mycmd=mycmd+'\r'
   ser.write(mycmd.encode())
```

2.i.

```
#include <Servo.h>
const int irPin1 = 7;
const int irPin2 = 8;

Servo gateServo; // define servo

void setup() {
Serial.begin(9600);
pinMode(irPin1, INPUT);
pinMode(irPin2, INPUT);
gateServo.attach(6);
gateServo.write(0); // set servo to initial position
}

void loop(){
int ir1State = digitalRead(irPin1);
int ir2State = digitalRead(irPin2);
if (ir1State == LOW) { // low indicates object detected
Serial.println("gate opening for approaching car");
gateServo.write(90);
}
if (ir2State == LOW) {
Serial.println("Gate closing for departing car");
gateServo.write(0);
}
// check for commands from serial interface
if (Serial.available() >0) {
String command = Serial.readStringUntil('\n');
command.trim() // remove trailing newline characters
if (command == "open") {
    gateServo.write(90);
    Serial.println("Gate manually opened");
} else if (command == "close") {
    gateServo.write(0);
    Serial.println("Gate manually closed");
} }
delay(100);
}
```

11. install pyserial library using pip install pyserial

```
import time
import serial
ser = serial.Serial('COM13', 9600) # Initialize serial communication (adjust port name and baud rate as needed)
time.sleep(2)
#function to update the sensor and servo status
def update_status():
    if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').strip()
        print(f"Status: {line}")

#function to manually open the gate
def open_gate():
    ser.write(b'open\n')
    print("Sent command to open the gate")

#function to manually close the gate
def close_gate():
    ser.write(b'close\n')
    print("Sent command to close the gate")

try:
    while True:
        update_status()
        user_input = input("Enter 'open' to open the gate, 'close' to close the gate, or 'exit' to quit: ").strip().lower()

        if user_input == 'open':
            open_gate() #Call function to open the gate
        elif user_input == 'close':
            close_gate() # call function to close the gate
        elif user_input == 'exit':
            break
        else:
            print("Invalid input")

except KeyboardInterrupt:
    print("Exiting the program")

ser.close()
```

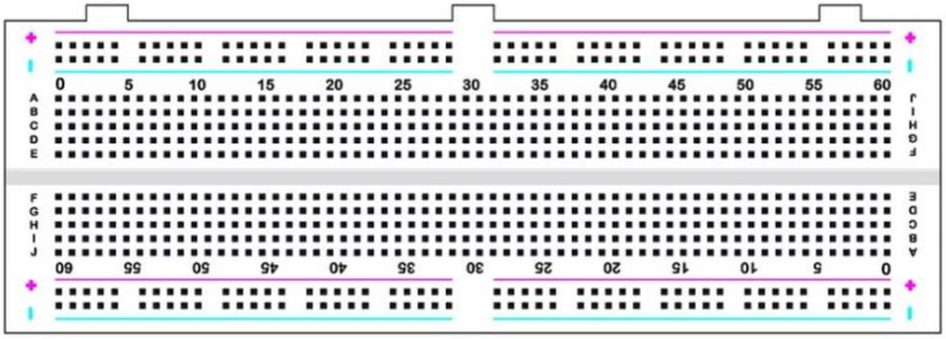
```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
//define pins for RFID
#define SS_PIN 10
#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN); // create MFRC522 instance
Servo drinkServo; // create servo instance

String validCards[] = {"12345678", "87654321"}; // example cards

void setup() {
 Serial.begin(9600); // initialize serial communication
 SPI.begin(); // initialize SPI bus
 mfrc522.PCD_Init(); // initialize MFRC522
 drinkServo.attach(6); // attach servo to pin 6
 drinkServo.write(0); // initilize servo position (closed)
 Serial.println("System initialized");
}

void loop() {
  if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) { // look for new cards
    delay(50);
    return;
  }
// read the card's UID
String cardID = "";
for (byte i = 0; i < mfrc522.uid.size; i++) {
   cardID += String(mfrc522.uid.uidByte[i], HEX);
}
cardID.toUpperCase(); // convert card ID to uppercase
```

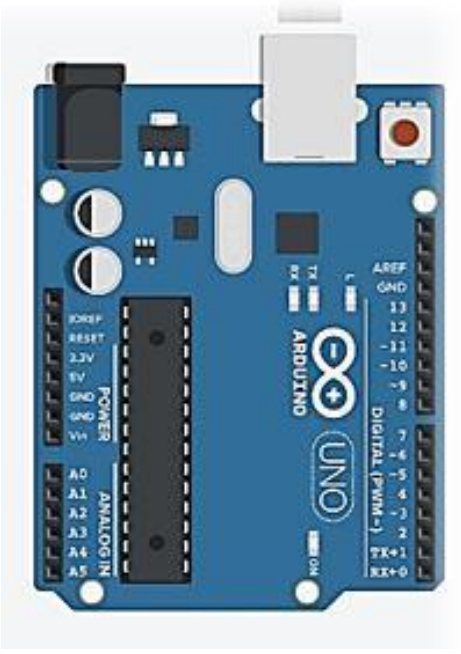
```
//check if card ID is valid
bool valid = false;
for (String id : validCards) {
  if (cardID == id) {
    valid = true;
    break;
  }}
//send status message to python script
if (valid) {
  Serial.println("Valid card detected: " + cardID);
}else {
  Serial.println("Invalid card detected: " + cardID);
  }

if (Serial.available()>0) {
  String command = Serial.readStringUntil('\n');
  command.trim();
  if (command == "renew "+ cardID) {
    Serial.println("Membership renewed for card: "+cardID)
    dispenseDrink();
  }}
delay(500);
}

void dispenseDrink(){
 drinkServo.write(90);
 delay(1000);
 drinkServo.write(0);
Serial.println("Drink dispensed");
}
```

```
import serial
import time

ser = serial.Serial('COM3', 9600) #initialize serial communication with the Arduino (adjust COM port and baud rate as needed)
time.sleep(2) #wait for the serial communication to initialize

def read_arduino():
    if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').strip() #read line, decode, and strip whitespace
        print(f"Arduino: {line}")
        return line
    return none

def renew_membership(card_id):
    command = f"renew {card_id}\n"
    ser.write(command.encode('utf-8') #send command to Arduino
    print(f"Sent command to renew membership for card: {card_id}")

try:
    while true:
        message = read_arduino() #read messages from arduino
        if message and "Valid card detected:" in message:
            card_id = message.split(": ")[1] #extract card ID from message
            print("Reminder: Please renew your membership.")
            user_input = input("Do you want to renew ur ---.? (yes/no): ").strip().lower()
            if user_input == 'yes':
                renew_membership(card_id) #send renewal command to arduino
except keyboardInterrupt:
    print("Exiting the program")

ser.close()
```

APPENDICES

APPENDIX 1 – Breadboard and Arduino Uno





APPENDIX 2 – Arduino Language Reference

Functions

For controlling the Arduino board and performing computations.

Digital I/O
digitalRead()
digitalWrite()
pinMode()

Analog I/O
analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family
analogReadResolution()
analogWriteResolution()

Advanced I/O
noTone()
pulseIn()
pulseInLong()
shiftIn()
shiftOut()
tone()

Time
delay()
delayMicroseconds()
micros()
millis()

Math
abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

Trigonometry
cos()
sin()
tan()

Characters
isAlpha()
isAlphaNumeric()
isAscii()
isControl()
isDigit()
isGraph()
isHexadecimalDigit()
isLowerCase()
isPrintable()
isPunct()
isSpace()
isUpperCase()
isWhitespace()

Random Numbers
random()
randomSeed()

Bits and Bytes
bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrupts
attachInterrupt()
detachInterrupt()

Interrupts
interrupts()
noInterrupts()

Communication
Serial
SPI
Stream
Wire

USB
Keyboard
Mouse

Structure

The elements of Arduino (C++) code.

Sketch

loop()
setup()

Control Structure

break
continue
do...while
else
for
goto
if
return
switch...case
while

Further Syntax

#define (define)
#include (include)
/* */ (block comment)
// (single line comment)
; (semicolon)
{} (curly braces)

Arithmetic Operators

% (remainder)
* (multiplication)
+ (addition)
- (subtraction)
/ (division)
= (assignment operator)

Comparison Operators

!= (not equal to)
< (less than)
<= (less than or equal to)
== (equal to)
> (greater than)
>= (greater than or equal to)

Boolean Operators

! (logical not)
&& (logical and)
|| (logical or)

Pointer Access Operators

& (reference operator)
* (dereference operator)

Bitwise Operators

& (bitwise and)
<< (bitshift left)
>> (bitshift right)
^ (bitwise xor)
| (bitwise or)
~ (bitwise not)

Compound Operators

%= (compound remainder)
&= (compound bitwise and)
*= (compound multiplication)
++ (increment)
+= (compound addition)
-- (decrement)
-= (compound subtraction)
/= (compound division)
^= (compound bitwise xor)
|= (compound bitwise or)

Variables

Arduino data types and constants.

Constants

HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED_BUILTIN
true | false
Floating Point Constants
Integer Constants

Conversion

(unsigned int)
(unsigned long)
byte()
char()
float()
int()
long()
word()

Data Types

array
bool
boolean
byte
char
double
float
int
long
short
size_t
string
String()
unsigned char
unsigned int
unsigned long
void
word

Variable Scope & Qualifiers

const
scope
static
volatile

Utilities

PROGMEM
sizeof()

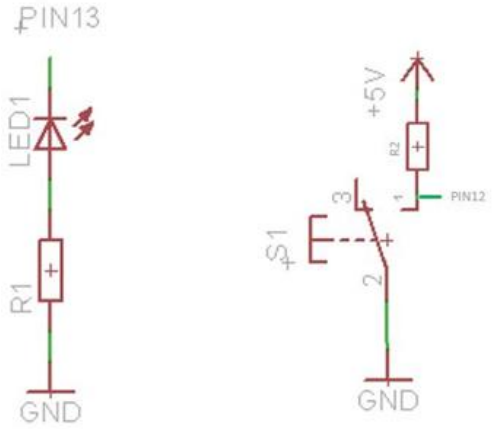
Examples

Writing digital output

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Reading digital input

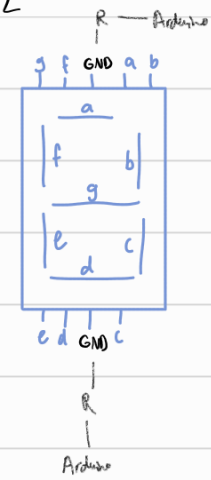
```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, INPUT);
  Serial.begin(9600);
}

void loop() {
  int state = digitalRead(12);
  Serial.println(state); // show state value on serial monito

  if (state != HIGH) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```



APPENDIX 3 – PYTHON PROGRAM STRUCTURE FOR USER INTERFACE

```
# import some libraries
import …

# Open the serial connection
…

# define routines that will be called from main()
def yourRoutines():
     …

def main():
    try:
        while True:
           …

    except KeyboardInterrupt:
           …

    # Close the serial connection
    finally:
           …

if __name__ == "__main__":
    main()
```

Week 2



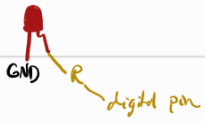
a b c d e f g ↔ digital pins 2 ~ 8


```
bool bPress = false;
int buttonPushCounter = 0;
int buttonState = 0;
int lastButtonState = 0;
void setup() {
pinMode(buttonPin, INPUT_PULLUP);
displaydigit(buttonPushCounter);
}
void loop() {
buttonState = digitalRead(buttonPin);
if (buttonState != lastButtonState) {     //compare buttonState to its previous state
    if (buttonState == LOW) {
        bPress = true;
        buttonPushCounter++;     // if state has changed, increment counter
        if (buttonPushCounter > 9) buttonPushCounter = 0;
        Serial.println("On");
    }
    else {
        Serial.println("Off");
    }
    delay(500);
}
```

```
lastButtonState = buttonState;
if (bPress) {
  turnoff();
  displaydigit(buttonPushCounter);
 }}
void displaydigit(int digit){
  if (digit != 1 && digit !=4)
  digitalWrite(a, HIGH);
  if (digit !=5 && digit !=6)
  digitalWrite(b, HIGH)
  if (digit !=2)
  digitalWrite(C, HIGH);
  if (digit !=1 && digit !=4 && digit !=7)
  digitalWrite(d, HIGH)
  if (digit ==2 || digit ==6 || digit ==8 || digit ==0)
  digitalWrite(e, HIGH);
  if (digit !=1 && digit !=2 && digit !=3 && digit !=7)
  digitalWrite(f, HIGH)
  if (digit !=0 && digit !=1 && digit !=7)
  digitalWrite(g, HIGH)
}
```

Week 3

POT



Try:

```
While True:
  pot_value = ser.readline().decode().strip() # assigns what on serial monitor to pot_value
  print("Potentiometer Value:", pot_value)
```

Servo

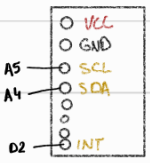
```
void loop (){
  if (Serial.available () >0) {
    int angle =Serial.parseInt ();  // read angle data from serial port
    servoMotor.write (angle);
  } }
```

```
import serial
import time
import sys
ser = serial.Serial ('COM7', 9600)
def move_servo(angle) :
    ser.write (str(angle).encode())
    time.sleep (0.1)

if __name__ == "__main__":
    print ("Enter an angle (0-180) or 'q' to quit.")
    While True:
        try :
            user_input = input ("Angle or 'q' to quit: ")
            if user_input.lower () =='q' :
                print("Exiting.")
                sys.exit ()
            angle = int (user_input)
            if 0 <= angle <=180 :
                move_servo (angle)
            else
```

```
//map potentiometer value (0, 1023) to servo angle (0, 180)
angle = map(potValue, 0, 1023, 0, 180);

if (Serial.available() > 0) {
  char receivedchar = Serial.read();
  if (receivedchar == 's') {
  servoMotor.write(90);
  }
}
```

```
import serial
import time
import sys
import keyboard
ser = serial.Serial('COM7', 9600)
def move_servo(angle):
    ser.write(str(angle).encode())
    time.sleep(0.1)

if __name__ == "__main__":
print("Use the potentiometer to adjust the servo angle.")
print("Press 'q' to quit, and 's' to stop the servo.")
try:
   while True:
        if keyboard.is_pressed('q'):
        print("Exiting program.")
        sys.exit()
        if keyboard.is_pressed('s'):
        print("stopping servo.")
        ser.write(b's') # send s
        time.sleep(0.1)
        pot_value = int(ser.readline().decode().strip())
        angle = int(map(pot_value, 0, 1023, 0, 180))
        move_servo(angle)
    except ValueError:
        print("Invalid input.")
```

Week 4

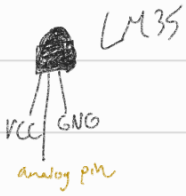



```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup() {
Serial.begin(9600);
Wire.begin();
mpu.initialize();
}

void loop(){
int16_t ax, ay, az, gx, gy, gz;
mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
Serial.print(ax); Serial.print(", ");
Serial.print(ay); Serial.print(", ");
Serial.print(az); Serial.print(", ");
Serial.print(gx); Serial.print(", ");
Serial.print(gy); Serial.print(", ");
Serial.print(gz); Serial.print(", ");
delay(100);
}
```

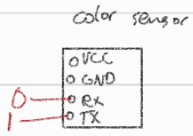
Python

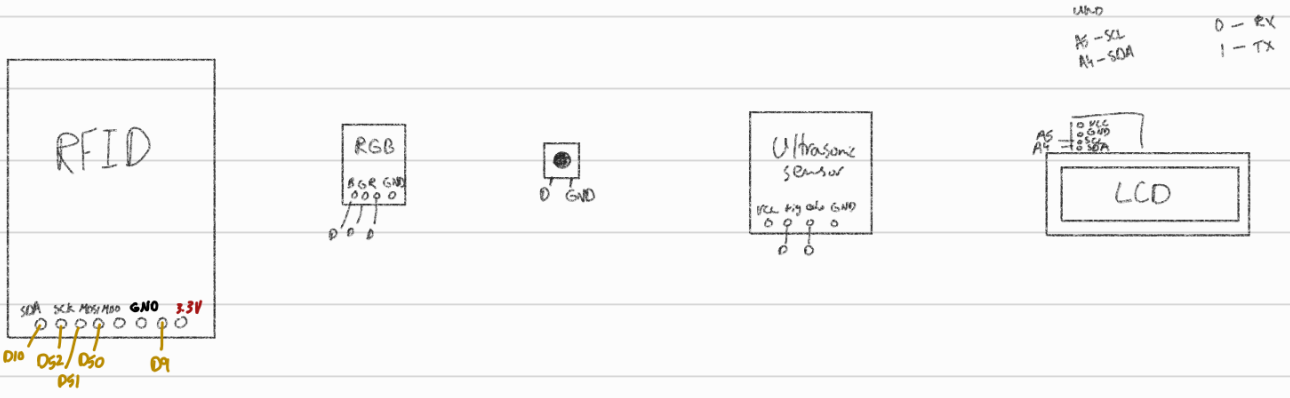
```
def read_sensordata():
    data = ser.readline().decode().strip()
    values = data.split(',')
    values = [int(val) for val in values]
    return values

if __name__ == "__main__":
    print("Ready sensor data from Arduino...")
    try:
        while True:
            sensor_data = read_sensordata()
            print("Accelerometr (x,y,z): ",sensor_data[:3])
            print("Gyroscope (x,y,z): ", sensor.data[3:])
            time.sleep(1)
except KeyboardInterrupt:
    print("Program terminated by user.")
    ser.close()
```

Week 6



Week 9






```
#include <LiquidCrystal_I2C.h>
#include <MFRC522.h>
MFRC522 mfrc522(10, 9)
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup(){
 mfrc522.PCD_Init();
 lcd.Init();
 lcd.backlight();
}
 void loop(){
 if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {    // to look for cards
 lcd.clear();
 lcd.print("What service?");
 lcd.setCursor(0, 1);
lcd.print("1: Wash 2: Dry");
}
}
Void setRGB(int red, int green, int blue) {
  analogWrite(RED_PIN, red);
  analogWrite(GREEN_PIN, green);
  analogWrite(BLUE_PIN, blue);
 }
```

```
import matplotlib.pyplot as plt
import numpy
import serial
from drawnow import
import matplotlib.animation as animation

potVal=[]
arduino=serial.Serial('COM7',9600)
plt.ion()

def makefig():
  plt.ylim(0,5)
  plt.title('Potentiometer values')
  plt.grid(True)
  plt.plot(potVal, 'ro-', label=' voltages')

try:
      while True:
            while(arduino.inWaiting()==0):
               pass
             arduinostring= arduino.readline()
             pot=float(arduinostring)
             potVal.append(pot)
             drawnow(makefig)
             plt.pause(0.000001)
             int cnt=cnt+1
             if(cnt>50):
                  potVal.pop(0)
```