

PROGRAM : BACHELOR ENGINEERING TECHNOLOGY
ELECTRICAL ENGINEERING

SUBJECT : DIGITAL TECHNOLOGY A2

CODE : DIGELA2

DATE : MID YEAR EXAMINATION
25 MAY 2019

DURATION : 12:30 – 15:30

WEIGHT : 40:60

TOTAL MARKS : 100

EXAMINER : MR. D.R. VAN NIEKERK 720011220

MODERATOR : JANE-ANNE BUISSON-STREET

NUMBER OF PAGES : 3 PAGES AND 2 ANNEXURES

INSTRUCTIONS

1. 100 MARKS = 100%. TOTAL MARKS AVAILABLE = 100
 2. ATTEMPT ALL QUESTIONS.
 3. ALL DIAGRAMS AND SKETCHES MUST BE DRAWN NEATLY AND IN PROPORTION.
 4. ALL DIAGRAMS AND SKETCHES MUST BE LABELLED CLEARLY.
 5. ALL WORK DONE IN PENCIL, EXCEPT DIAGRAMS AND SKETCHES, WILL BE CONSIDERED AS ROUGH WORK AND WILL NOT BE MARKED.
 6. MARKS WILL BE DEDUCTED FOR WORK THAT IS POORLY PRESENTED.
 7. QUESTIONS MAY BE ANSWERED IN ANY ORDER, BUT ALL PARTS OF A QUESTION, MUST BE KEPT TOGETHER.
 8. ONLY ONE POCKET CALCULATOR PER CANDIDATE MAY BE USED.
-



QUESTION 1

- 1.1 Explain why a physical press of the on-board reset button is not required when unloading code to the Arduino Uno? (4)
- 1.2 Explain how the Arduino Uno board protects the computer connected USB port from a short or overload? (4)
- 1.3 Explain the operational difference between a “while” loop and a “do...while” loop? (3)
- 1.4 Consider the “break” and “continue” statements. Where should each one be used, and what is the operational difference? (5)
- [16]
-

QUESTION 2

- 2.1 What is the meaning of a “LOW” in reference to a port pin when it is configured as an “INPUT” and as an “OUTPUT” on the Arduino board? (6)
- 2.2 When Arduino port pins are configured as “OUTPUT”, what state are they in and what can they provide to other external circuits? (3)
- 2.3 Describe the “const” keyword, what it does and the benefit of using it. (5)
- 2.4 What is the purpose of using an interrupt and name two tasks that an interrupt may be used for? (5)
- [19]
-

QUESTION 3

- 3.1 Describe and explain the operation of the Arduino “pulseIn()” function and its range? (6)
- 3.2 Write an Arduino C program using the “pulseIn()” function to switch on a relay controlled by digital pin 4, if a low pulse width on input pin 3 is greater than one milliseconds. The Arduino must wait for at least five seconds to detect the low pulse width. If the pulse width becomes less than or equal to one millisecond or is not detected, the relay must switch off. (8)
- 3.3 Write an Arduino C program to serial print at 9600 baud rate, the analog value on pin “A0” every second. Print tab spaced on each line, the analog result as a decimal, hexadecimal and a binary value. (9)
- [23]

QUESTION 4

- 4.1 Without component values, sketch a basic P-channel MOSFET switch for a 12Vdc and a 24Vdc supply, “sourcing” a lamp load. Explain why a Zener diode is required for the 24Vdc supply and what the configuration is often called? (9)
- 4.2 Sketch an optically-isolated resistive triac switch circuit connected to an Arduino digital output pin which is used to control a 230V AC lamp load. Calculate the LED limiting resistor for the MOC3022 optically-isolated triac driver ($I_f = 10\text{mA}$, $V_f = 1.5\text{V}$) used to drive the BTA06-600C non-sensitive gate power triac. (6)
- 4.3 Give five optically isolated triac device applications. (5)
- [20]
-

QUESTION 5

- 5.1 Write an Arduino C program that initially displays “Scroll” fixed text on the first LCD line. Then every half (1/2) second scrolls the display right ten times and then every half (1/2) second scrolls the display left ten times and repeats this operation continuously. Only four LCD data lines are connected to Arduino pins 4 to 7. The LCD register select is connected to pin 8 and the LCD enable line is connected to pin 9. A sixteen (16) by two-line LCD is used. (11)
- 5.2 How long does it take to write a byte to EEPROM and how many times can it be written? (2)
- 5.3 What is the Arduino “servo.writeMicroseconds()” library function used for and describe how it effects a standard and continuous rotation servo. (9)
- [22]
-

TOTAL [100]

Arduino Functions:

pinMode(<i>pin</i> , <i>mode</i>)	Serial.write(buf, len)
digitalWrite(<i>pin</i> , <i>value</i>)	Serial.parseInt()
digitalRead(<i>pin</i>)	Serial.parseInt(char skipChar)
analogReference(<i>type</i>)	Serial.parseFloat()
analogRead(<i>pin</i>)	Serial.readBytes(buffer, length)
analogWrite(<i>pin</i> , <i>value</i>)	Serial.readBytesUntil(terminator, buffer, length)
tone(<i>pin</i> , frequency)	Serial.readString()
tone(<i>pin</i> , frequency, duration)	Serial.readStringUntil(terminator)
noTone(<i>pin</i>)	Serial.find(target)
shiftOut(<i>dataPin</i> , <i>clockPin</i> , <i>bitOrder</i> , <i>value</i>)	Serial.findUntil(target, terminator)
shiftIn(<i>dataPin</i> , <i>clockPin</i> , <i>bitOrder</i>)	Serial.setTimeout(time)
pulseIn(<i>pin</i> , <i>value</i>)	serialEvent()
pulseIn(<i>pin</i> , <i>value</i> , timeout)	LiquidCrystal lcd(rs, en, d4, d5, d6, d7)
millis()	LiquidCrystal lcd(rs, rw, en, d0, d1, d2, d3, d4, d5, d6, d7)
micros()	lcd.begin(cols, rows)
delay(ms)	lcd.clear()
delayMicroseconds(us)	lcd.home()
min(x, y)	lcd.setCursor(col, row)
max(x, y)	lcd.write(byte)
abs(x)	lcd.print(data)
constrain(num, min, max)	lcd.print(data, base)
map(Xin, Xmin, Xmax, Ymin, Ymax)	lcd.cursor()
pow(base, exponent)	lcd.noCursor()
sqrt(num)	lcd.blink()
sin(rad)	lcd.noBlink()
cos(rad)	lcd.display()
tan(rad)	lcd.noDisplay()
randomSeed(seed)	lcd.scrollDisplayLeft()
random(max)	lcd.scrollDisplayRight()
random(min, max)	lcd.leftToRight()
lowByte(x)	lcd.rightToLeft()
highByte(x)	lcd.autoscroll()
bitRead(x, n)	lcd.noAutoscroll()
bitWrite(x, n, b)	lcd.createChar(num, data)
bitSet(x, n)	EEPROM.read(address)
bitClear(x, n)	EEPROM.write(address, value)
bit(n)	EEPROM.update(address, value)
attachInterrupt(interrupt, ISR, mode)	EEPROM.put(address, data)
detachInterrupt(interrupt)	EEPROM.get(address, data)
noInterrupts()	EEPROM[address]
Interrupts()	servoM1.attach(pin)
Serial.begin(speed)	servoM1.attach(pin, min, max)
Serial.begin(speed, config)	servo.write(angle)
Serial.end()	servo.writeMicroseconds(us)
Serial.available()	servo.read()
Serial.flush()	servo.attached()
Serial.print(val)	servo.detach()
Serial.print(val, format)	NumKeys.begin(makeKeymap(keys));
Serial.println(val)	NumKeys.setDebounceTime(time);
Serial.println(val, format)	NumKeys.setHoldTime(time);
Serial.read()	char ckey = NumKeys.getKey()
Serial.peek()	byte State = NumKeys.getState()
Serial.write(val)	boolean State = NumKeys.keyStateChanged()
Serial.write(str)	char ckey = NumKeys.waitForKey()
	NumKeys.addEventListener(keypadEvent)

Constants:

true, false, HIGH, LOW, INPUT, INPUT_PULLUP, OUTPUT, LED_BUILTIN, DEFAULT, INTERNAL, EXTERNAL, MSBFIRST, LSBFIRST, CHANGE, RISING, FALLING, BIN, OCT, DEC, HEX, PRESSED, HOLD, RELEASED, IDLE

Data Types:

boolean : 0 to 1 (false or true)
 char or signed char : -128 to 127 (sign bit plus 7 data bits)
 byte or unsigned char : 0 to 255 (8 data bits)
 int or signed int or short : -32768 to 32767 (sign bit plus 15 data bits)
 word or unsigned int : 0 to 65535 (16 data bits)
 long or signed long : -2E31 to 2E31-1 (sign bit plus 31 data bits)
 unsigned long : 0 to 2E32-1 (32 data bits)
 float or double : -3.4E38 to 3.4E38 (32 data bits) Arduino ATmega based boards

Data Qualifiers:

void : No qualified data type
 auto : Ram variables where memory is re-used once the function is executed (default)
 static : RAM variables that stay intact once function is executed
 volatile : RAM variables that are changeable by background routines (interrupt routines)
 const : ROM space reserved for fixed values

Operators: (High to low priority)

() [] : Parenthesized Expression, Array Subscript
 . -> : Structure Member and Pointer
 + - : Unary + and – (Positive and Negative Signs)
 ++ -- : Increment and Decrement
 ! ~ : Logical NOT and Bitwise Complement
 * & : Dereference (Pointer) and Address of
 sizeof (type) : Size of Expression or Type and Explicit Typecast
 * / % : Multiply, Divide, and Modulus
 + - : Add and Subtract
 << >> : Shift Left and Shift Right
 < <= : Less Than and Less Than or Equal To
 > >= : Greater Than and Greater Than or Equal To
 == != : Equal To and Not Equal To
 & ^ | : Bitwise AND, XOR and OR
 && || : Logical AND, Logical OR
 ?: = : Conditional Operator and Assignment Operator
 += -= : Addition and Subtraction Assignments
 /= *= : Division and Multiplication Assignments
 %= : Modulus Assignment
 <<= >>= : Shift Left and Shift Right Assignments
 &= |= ^= : Bitwise AND, OR, and XOR Assignments
 , : Comma Operator

Structures:

void setup () {...}
 void loop () {...}
 if (...) {...} else {...}
 for (...; ...; ...) {...}
 while (...) {...}
 do {...} while (...);
 switch (...) {case .. : ... break; case .. : ... break; default: ...}
 break; continue; goto ..; return or return ..; //... or /*...*/ for comments
 #define & #include <...> or #include "..."