Electronic Supplementary Material (ESI) for Reaction Chemistry & Engineering.

A tutored discourse on microcontrollers, single board computers and their applications to monitor and control chemical reactions

Daniel E. Fitzpatrick, Matthew O'Brien and Steven V. Ley

Appendix A – Item list for workshop

Mini-Breadboard:
https://www.amazon.co.uk/gp/product/B01N0YWIR7/

10k Ω Potentiometer:
https://www.amazon.co.uk/gp/product/B00JR6HZLK/

Momentary Push-Button Switch:
https://www.amazon.co.uk/gp/product/B071RBCRVH/

ESP32 Microcontroller Board:
https://www.amazon.co.uk/gp/product/B078CXRNP9/

Breadboard Jumper Wire Pack:
https://www.amazon.co.uk/gp/product/B06Y2JHGGV/

220 Ω Resistors:
https://www.amazon.co.uk/gp/product/B00YW0HLP6/

4k7 Ω Resistors:
https://www.amazon.co.uk/gp/product/B00UBUB76U/

28BYJ-48 Stepper Motor and Driver Board:
https://www.amazon.co.uk/gp/product/B01IP7IOGQ/

LEDs:
https://www.amazon.co.uk/gp/product/B07Q1813PN

6-Way USB Charger/Power Supply:
https://www.amazon.co.uk/gp/product/B07CXJ7B9G

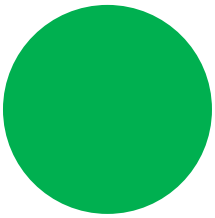
Raspberry Pi 3 Model B+:
https://uk.rs-online.com/web/p/products/1373331/

Raspberry Pi Power Supply:
https://uk.rs-online.com/web/p/products/9098126/

Raspberry Pi Micro-SD Card 8GB:
https://uk.rs-online.com/web/p/micro-sd-cards/6957334/

USB-RS232 Serial Adapter:
https://www.lindy.co.uk/usb-thunderbolt-c4/usb-converters-c213/usb-to-4-port-serial-converter-p511

Appendix B – Green dot for machine vision



Appendix C – CherryPy Full Code Example

```
import serial
import urllib
import cherrypy
import json
import os
from time import sleep

#Connect to serial port
ser = {}

class SerialServer(object):
    @cherrypy.expose
    def index(self):
        return "Server is running. Use /command?port=&command="

    @cherrypy.expose
    def initiateport(self, varport='ttyUSB0', varbaudrate=38400, vartimeout=0.5):
        try:
            ser[str(varport)] = serial.Serial(
                port='/dev/'+varport,
                baudrate=varbaudrate,
                parity=serial.PARITY_NONE,
                stopbits=serial.STOPBITS_ONE,
                bytesize=serial.EIGHTBITS,
                timeout=float(vartimeout)
            )
            return "ok"
        except:
            return "error"

    @cherrypy.expose
    def command(self, port=0, command="", equipmenttype=""):
        if(command == ''):
            return "Blank command"

        return executeCommand(port, command, equipmenttype)

def executeCommand(port, command, equipmenttype):
    global ser
    ser[str(port)].write(command.encode('latin-1'))
    sleep(0.05)
    returnedvalue = ser[str(port)].read(ser[str(port)].inWaiting())
    returnedvalue = returnedvalue.replace(b'\r',b'').replace(b'\n',b'')
    return returnedvalue

#Start server
cherrypy.config.update({'server.socket_port': 80})
cherrypy.config.update({'server.socket_host': '0.0.0.0'})
cherrypy.config.update({'server.thread_pool': 50})
cherrypy.quickstart(SerialServer(), '/')
```

Appendix D – Glossary: Some Useful Terms

Term	Description
AI	Artificial Intelligence
Algorithm	Sequence of instructions to solve a problem or to perform a calculation
Bootloader	Program that enables loading of the operating system on start-up
Code	Set of instructions forming a computer program
CPU	Central Processing Unit
DoE	Design of Experiments
ELN	Electronic Laboratory Notebook
Flash Memory	Non-volatile memory chip used for storage
GitHut	Collections of improved software for open source repositories
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
LED	Light Emitting Diode
ML	Machine Learning
Powershell	Allows system administrator and power-users to rapidly automate tasks that manage operating systems
RAM	Random Access Memory
REPL	Read-Evaluate-Print Loop
RS232	Interface between a data terminal and equipment using serial binary data exchange
Simplex	A statistical optimisation method
SNOBFIT	Stable Noisy Optimisation and FIT
Tool Chain	Set of programming tools to perform a complex software development task
USB	Universal Serial Bus

Appendix E – Code examples (text)

Block 1

```
#include <stdio.h>

main ()
{
  printf("hello FET");
}
```

Block 2

```
gcc -o myfile.out myfile.c
```

Block 3

```
./myfile.out
```

Block 4

```
objdump -f ./myfile.out
```

Block 5

```
objdump -d ./myfile.out
```

Block 6

```
objdump -x ./myfile.out
```

Block 7

```
xxd -b ./myfile.out
```

Block 8

```
gcc -S myfile.c
```

Block 9

```
#include <stdio.h>
#include <stdlib.h>

float ctof(int c)
{
  float f = (9.0*c/5.0)+32.0;
  return f;
}

main ()
{
  char i[6] = {'\0'};

  printf("enter a number, 5 digits or less: ");
  fgets(i, 6, stdin);

  int x;
  for(x=0; x<6; x++)
  {
    if (i[x] == '\n')
    {
      i[x] = '\0';
```

```
    }
  }

  printf("your string is: %s\n", i);
  int cel = atoi(i);
  printf("your integer is: %d\n", cel);
  printf("%d in fahrenheit = %f\n", cel, ctof(cel));
}
```

Block 10

```
#include <stdio.h>
#include <stdlib.h>
float ctof(int c){float f = (9.0*c/5.0)+32.0;return f;} main (){char i[6] =
{'\0'};printf("enter a number, 5 digits or less: ");fgets(i, 6, stdin);int
x;for(x=0; x<6; x++){if (i[x] == '\n'){i[x] = '\0';}}printf("your string is: %s\n",
i);int cel =atoi(i);printf("your integer is: %d\n", cel);printf("%d in fahrenheit =
%f\n", cel, ctof(cel));}
```

Block 11

```
#define MYPIN 1

void setup()
{
  pinMode(MYPIN, OUTPUT);
}


void loop()
{
  digitalWrite(MYPIN, HIGH);
  delay(400);
  digitalWrite(MYPIN, LOW);
  delay(400);
}
```

Block 12

```
int sensorPin = 34;
int sensorValue = 20;

int pin1 = 13;
int pin2 = 12;
int pin3 = 14;
int pin4 = 27;

int pins[] = {pin1, pin2, pin3, pin4};

const boolean stepseq[8][4] =
{
  {1,0,0,0},
  {1,1,0,0},
  {0,1,0,0},
  {0,1,1,0},
  {0,0,1,0},
  {0,0,1,1},
  {0,0,0,1},
  {1,0,0,1}
};


void setup()
{
  pinMode(pin1, OUTPUT);
```

```
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  Serial.begin(115200);
  Serial.println("starting");
}

void loop()
{
  for (int i=0; i<8; i++)
  {
    for (int j=0; j<4; j++)
    {
      Serial.print(stepseq[i][j]);
      digitalWrite(pins[j], stepseq[i][j]);
    }
    Serial.println("");
    sensorValue = analogRead(sensorPin);
    delay(1 + sensorValue/100);
  }
}
```

Block 13

```
import RPi.GPIO as gpio
from time import sleep
gpio.setmode(gpio.BOARD)
gpio.setup(3, gpio.OUT)
for x in range(10):
    gpio.output(3, 1)
    sleep(0.1)
    gpio.output(3, 0)
    sleep(0.1)
```

Block 14

```
import RPi.GPIO as gpio
from time import sleep

gpio.setmode(gpio.BOARD)

gpio.setup(3, gpio.OUT)
gpio.setup(5, gpio.OUT)
gpio.setup(7, gpio.OUT)

def blink(n):
    for x in range(10):
        gpio.output(n, 1)
        sleep(0.1)
        gpio.output(n, 0)
        sleep(0.1)

ledpins = {'green': 3, 'yellow': 5, 'red': 7}

while True:
    x = input("enter colour: ")

    if x in ledpins:
        blink (ledpins[x])

    else:
        print("enter either 'red', 'green' or 'yellow'")

    if x=='quit':
        break
```

Block 15

```
import RPi.GPIO as gpio
from time import sleep
import threading

gpio.setmode(gpio.BOARD)

gpio.setup(3, gpio.OUT)
gpio.setup(5, gpio.OUT)
gpio.setup(7, gpio.OUT)

def blink(n):
    for x in range(10):
        gpio.output(n, 1)
        sleep(0.5)
        gpio.output(n, 0)
        sleep(0.5)

while True:
    x = input("enter colour: ")

    ledpins = {'green': 3, 'yellow': 5, 'red': 7}

    if x in ledpins:
        thread1 = threading.Thread(target=blink, args=(ledpins[x],))
        thread1.start()

    else:
        print("enter either 'red', 'green' or 'yellow'")

    if x=='quit':
        break
```

Block 16

```
thread1 = threading.Thread(target=blink, args=(ledpins[x],))
thread1.start()
```

Block 17

```
import RPi.GPIO as gpio
from time import sleep
import threading
import queue

gpio.setmode(gpio.BOARD)

gpio.setup(3, gpio.OUT)
gpio.setup(5, gpio.OUT)
gpio.setup(7, gpio.OUT)

FINISH = False

ledpins = {'green': 3, 'yellow': 5, 'red': 7}

redq = queue.Queue()
greenq = queue.Queue()
yellowq = queue.Queue()

def blink(n):
    for x in range(10):
        gpio.output(n, 1)
        sleep(0.5)
        gpio.output(n, 0)
        sleep(0.5)
```

```
def redrunner():
    while True:
        if FINISH:
            print ('redrunner breaking')
            break
        sleep(0.0001)
        if not redq.empty():
            redq.get()
            blink(ledpins['red'])

def yellowrunner():
    while True:
        if FINISH:
            print('yellowrunner breaking')
            break
        sleep(0.0001)
        if not yellowq.empty():
            yellowq.get()
            blink(ledpins['yellow'])

def greenrunner():
    while True:
        if FINISH:
            print('greenrunner breaking')
            break
        sleep(0.0001)
        if not greenq.empty():
            greenq.get()
            blink(ledpins['green'])

redthread = threading.Thread(target=redrunner)
yellowthread = threading.Thread(target=yellowrunner)
greenthread = threading.Thread(target=greenrunner)

redthread.start()
yellowthread.start()
greenthread.start()

while True:
    x = input("enter colour: ")

    if x == 'red':
        redq.put(x)
        print('putting in redq')

    elif x == 'yellow':
        yellowq.put(x)
        print('putting in yellowq')

    elif x == 'green':
        greenq.put(x)
        print('putting in greenq')

    else:
        print("enter either red, green or yellow")

    if x=='quit':
        FINISH = True
        print('quitting')
        break
```

Block 18

```
def redrunner():
    while True:
        if FINISH:
            print ('redrunner breaking')
            break
```

```
        sleep(0.0001)
        if not redq.empty():
            command = redq.get()
        if command == 'red1':
             blink1(ledpins['red'])
        if command == 'red2':
             blink2(ledpins['red'])
```

Block 19

```
funcs = {'func1':func1, 'func2':func2, 'func3':func3}
```

Block 20

```
def redrunner():
    while True:
        if FINISH:
            print ('redrunner breaking')
            break
        sleep(0.0001)
        if not redq.empty():
            command = redq.get()
            thread1 = threading.Thread(target = funcs[command])
            thread1.start()
```

Block 21

```
thread1.join()
```

Block 22

```
# Initialise and connect to web camera

# Loop to capture position of green dot at regular intervals

     # Get image from camera

     # Locate position of green dot

# Display position coordinates
```

Block 23

```
# Initialise and connect to web camera

from time import sleep
from SimpleCV import *
cam = Camera()
```

Block 24

```
# Loop to capture position of green dot at regular intervals

while True:

       # Get image from camera

       # Locate position of green dot

       # Display position coordinates

sleep(0.25)
```

Block 25

```
        # Get image from camera

        image = cam.getImage()
```

Block 26

```
        # Locate position of green dot

        image = image.toRGB()
        image = image.splitChannels()

        imageR = image[0]
        imageG = image[1]
        imageB = image[2]
```

Block 27

```
        imageGminB = imageG-imageB
        imagebinarize = imageGminB.binarize()
        imageinvert = imageGminB.invert()
```

Block 28

```
        blobs = imageGminB.findBlobs()
```

Block 29

```
      # Display position coordinates

      print blobs[0].centroid()
```

Block 30

```
import serial
import urllib
import cherrypy
import json
import os
```

Block 31

```
#Connect to serial port

ser = serial.Serial(
    port='/dev/ttyUSB0',
    baudrate=19200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_TWO,
    bytesize=serial.EIGHTBITS,
    timeout=0.5
)
```

Block 32

```
class SerialServer(object):
    @cherrypy.expose
    def index(self):
        return "Server is running"

    @cherrypy.expose
    def initiateport(self, varbaudrate=19200, vartimeout=0.5):
        try:
           ser.close();
```

```
            ser = serial.Serial(
                port='/dev/ttyUSB0',
                baudrate=varbaudrate,
                parity=serial.PARITY_NONE,
                stopbits=serial.STOPBITS_ONE,
                bytesize=serial.EIGHTBITS,
                timeout=float(vartimeout)
            )
            return "ok"
        except:
            return "error"

    @cherrypy.expose
    def command(self, command=""):
        global ser

        if(command == ''):
            return "Blank command"

        ser.write(command.encode('latin-1'))
        returnedvalue=ser.read(ser.inWaiting())
        return returnedvalue.replace(b'\r',b'').replace(b'\n',b'')
```

Block 33

```
#Start server
cherrypy.config.update({'server.socket_port': 80})
cherrypy.config.update({'server.socket_host': '0.0.0.0'})
cherrypy.config.update({'server.thread_pool': 50})
cherrypy.config.update(SerialServer(), '/')
```