Barry the Plant Watering Robot

We are going to create the controller board for Barry, a robot who will do some of our plant watering chores for us. We aren't going to build all of Barry, but this will be his brain.

Set up the LED screen

Goals

- ❏ Initiate the LCD screen
- ❏ Create a function that displays text on the screen the way we want
- ❏ Understand the limitations of the LCD screen (how many lines, how many characters)
- ❏ Understand how to print and clear quickly by creating a function that calls a function

1. The top of the file lists all of the imports and initialisations required by Barry.

2. These lines initialize the LCD screen. They should already be there:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(6, 7, 8, 2, 3, 4, 5);
```

3. To test: Under **void setup()**:

```
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("hello, world!");
```

4. Create a function that will print to the LCD in a single line:

```
void printLCD(String row1, String row2) {
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print(row1);
  lcd.setCursor(0, 1);
  lcd.print(row2);
  delay(waitTime/6);
}
```

5. To test: Under **void setup()** pass some values to your new printLCD function.

6. Now… USE the **printLCD**() function to make a clearLCD function. Don't copy the same code over though, actually use the printLCD function in your clearLCD function (aka funtception).

7. Test your new functions under **void setup()**.

Set up Day and Night

Goals

- ❏ Use the light sensor to simulate the time passing to the next day
- ❏ Reflect the new day on the LED indicator
- ❏ Display the new day on the LCD

1. In the **initialisation area**, these are the values that initialize the LED register. They should already be there for you:

```
const int shiftLatchPin = 11;
const int shiftDataPin = 12;
const int shiftClockPin = 10;
```

2. These should already exist for you under **void setup().** They activate the LED register for our program:

```
pinMode(shiftLatchPin, OUTPUT);
pinMode(shiftDataPin, OUTPUT);
pinMode(shiftClockPin, OUTPUT);
```

3. A function to *use* the LED register already exists. Check it out. We will pass a number to it to make it work:

```
void displayBinary(int number) {
      digitalWrite(shiftLatchPin, LOW);
      shiftOut(shiftDataPin, shiftClockPin, LSBFIRST, number);
      digitalWrite(shiftLatchPin, HIGH);
}
```

4. The idea is that we will pass the number to the function to refer to the day of the week, and it will count up the days for us. Therefore we need to find the numbers that represent just one light on at a time. The LED register works on binary numbers. Where there is a 1 in the binary, the light is on, where there is a 0, the light will be off. Test numbers under **void setup()** using the **displayBinary** function to work out which binary numbers represent each light. We will only need 7 lights.

5. We can set up an ENUM to help us refer to the days of the week instea of numbers when using the LED register.

   Some of the ENUM is already typed in the **variable declaration area**. Fill in the blanks.

```
enum WeekDays {
```

```
            Monday = ?,
            Tuesday = ?,
            Wednesday = ?,
            Thursday = ?,
            Friday = ?,
            Saturday = ?,
            Sunday = ?
                  };
```

6. Also add a day tracker to store the enum value we're using at that moment:

```
      int currentDay;
```

7. Test your code under **void setup(**). Pass values to the LED register using your **displayBinary** function and the ENUM.

8. Experiment yourself to find the value for NO lights at all. Include this code in your **void setup()** after the pinMode lines.

9. After the line making sure all the lights are off, add:

```
      currentDay = Monday;
```

10. Now we will set up the day and night cycle. It will be simulated with the light sensor. Go to the **void loop()** part of the code and add:

```
      int lightSensor = analogRead(7);
```

11. Then we add the code to make it change days when 'night' comes.

```
      if( lightSensor < 30 ) {
            if( currentDay == Sunday ) {
                  currentDay = Monday;
            } else {
                  currentDay /= 2;
            }
            //printLCD("It's now ",weekDay(currentDay));
            displayBinary(currentDay);
            delay(waitTime);
      }
```

30 is the value that has been chosen by trial and error. If you change this value it will be easier or harder to make it become 'night'.

12. At this point we should also make the reverse of the ENUM. In this case we are making a Switch statement to turn numbers into Strings. Part of this function is typed for you, but you need to complete the correct syntax.

```
String weekDay(int binaryNum) {
  switch (binaryNum){
    case 128:
    return "Monday";
    case 64:
    return "Tuesday";
    case 32:
    return "Wednesday";
    case 16:
    return "Thursday";
    case 8:
    return "Friday";
    case 4:
    return "Saturday";
    case 2:
    return "Sunday";
      }
}
```

13. Test your new day and night code. Running the application, then putting your finger over the light sensor should make the days increment.

14. Also you can now uncomment the line `printLCD("It's now",weekDay(currentDay));` With your new function **weekDay**, it will work.

Plant Stuff

Goals
- ❏ Prompt for the plants we have
- ❏ Using the buttons, answer yes or no questions about the plants
- ❏ Establish how much time they need between watering

1. Set up at least TWO lots of variables in the **initialization area** for our plants.
   a. Boolean for whether or not we have the plant - Barry will ask.
   b. String for the name of the plant - set this value yourself.
   c. One set of integers for the number of days until they require watering - set this value yourself.
   d. One set of integers for the number of days it has been since they were watered - Barry will calculate.

2. Add initialisation for the buttons because we will use these to answer questions:

```
const int yesButtonPin = 15;       // to register a yes
const int noButtonPin = 14;        // to register a no
```

3. Set up a function that asks us if we have each plant.

```
boolean flowerAsk (String flowerType){

  printLCD("Type: " + flowerType + "?","No           Yes");
  delay(waitTime/10);

  int buttonYesPressed;
  int buttonNoPressed;
  digitalWrite(yesButtonPin, HIGH);
  digitalWrite(noButtonPin, HIGH);

while (true) {

  buttonYesPressed = digitalRead(yesButtonPin);
  buttonNoPressed = digitalRead(noButtonPin);

    if (buttonYesPressed == LOW) {
      clearLCD();
      printLCD("Yes","");
```

```
        return true;
      }

      if (buttonNoPressed == LOW) {
        clearLCD();
        printLCD("No", "");
        return false;
      }
    }
  }
```

4. Using this function, in the **void setup()** area, prompt to ask if we have each kind of plant. Record the answer in your Boolean parameter for each plant.

Set up the Temperature

Goals
- ❏ Use the potentiometer to simulate the temperature today
- ❏ Make the potentiometer less sensitive - it is not going to be 250C

1. In the **initialization area** add the potentiometer:

```
      const int potPin = 3;   // the number of the potentiometer pin
      int potValue = 0;       // variable for reading the potentiometer
value
      int todaysTemp;        // variable to store the temperature today
```

2. Create a function that reads the potentiometer and makes it less sensitive.

```
      int temperatureValue() {
            int countDown = 12;
            clearLCD;
            while (0 < countDown) {
                  potValue = (analogRead(potPin))/16; // divide by 16 so that the
                  max value is not hotter than Venus
                  printLCD("Today's temp:",String(potValue) + " degrees");
                  countDown--;
                  }
            return potValue;
            }
```

3. Test this function under **void setup()**

Deal with a New Day

Goals

- ❏ Barry recognises when it is a new day and will check to see what plants need watering
- ❏ Checks the temperature so that all plants get watered if it's really hot

1. Consider what has to happen for Barry to work:
   a. He needs to know what plants we have, and how long between watering they can wait - DONE
   b. He needs to know that a new day has dawned so he can checked when the plants were last watered, and check when they need to be watered - DONE
   c. He needs to know if it's too hot, meaning all plants get watered - DONE

2. The final step is to put create a function where Barry runs through his daily plant watering responsibilities. Therefore, create a function called **void plantloop()** that runs when a new day dawns. It must:
   a. Use the potentiometer to test the temperature. If it is too hot, all the plants must be watered and their days since last watered counter set to zero.
   b. If it is not too hot, Barry has to check how long since each of the plants was last watered, and if it has hit the limit, water them.
   c. Barry should tell us at each stage what he is doing.

3. This function should trigger INSIDE the light sensor loop, after we increment the day by one.

```
void plantLoop(){
      // test for temperature here, store in a variable
      delay(waitTime/2);

      if (it's too hot today) {
            //display what is happening on the LCD
            //set the days since last watered variables for all plants
            delay(waitTime);
            }

      if (// check if you have Plant1){
            if (//check if it is time to water plant1 based on
      days elapsed since last watering){
        //display what is happening on the LCD
        // set the days since last watered variables
        delay(waitTime);
} else {
        //display what is happening on the LCD
        // set the days since last watered variables
        delay(waitTime);
        }
}

...
//repeat for the other plant variables you have (plant2,
plant3, plant4..)
…

}
```

Cleanup

Goals

- ❏ Run your code and test what is happening
- ❏ Some things are missing - what are they,
- ❏ What can you add? Barry could do heaps of things

1. Some things are missing. Look carefully at the day's counter. Shouldn't Barry know how long it has been since the plants were watered? Where would you put this?

2. What about the first time Barry boots? Shouldn't he run through his plant watering duties even on the first day? Create a way for Barry to know it is the first day and run through the **plantloop** function.

3. Other things to try - make a plant die if the temperature is 0 degrees. Make it not exist in the loop anymore... Its dead Jim.

4. You could create a mechanism with the potentiometer to record the alphabet. Create a character array using **char alphabet[26]={'A','B'....}** and then you could create a prompt with the buttons for which kinds of plants you have. You could change Barry's name. If you did the same with numbers, on the initial loop, you could set the number of days a plant requires to be watered yourself.

Extra - AntiTheft Device

Goals

- ❏ Engage the accelerometer to trigger if Barry is moved

The accelerometer is a cool but tricky device. We are going to engage it in the most simplistic manner to illustrate its potential. If the board is moved too much, it will trigger a help message.

1. In the **initialization area** include these libraries and variables:

```
#include <Wire.h>       //Include the Wire library
#include <MMA_7455.h>   //Include the MMA_7455 library
const int xpin = 12;    // x-axis of the accelerometer
const int ypin = 12;    // y-axis
const int zpin = 12;    // z-axis (only on 3-axis models)
char xVal, yVal, zVal;  // Return value variables for the
accelerometer
```

2. These will do the basic setup for the accelerometer under **void setup():**

```
        Serial.begin(9600);
        accel.initSensitivity(2);
        accel.calibrateOffset(0, 0, 63);
```

3. Create a **void noSteal()** function:

```
      void noSteal() {

        // Get the X, Y, anx Z axis values from the device
        xVal = accel.readAxis('x');   // Read X Axis
        yVal = accel.readAxis('y');   // Read Y Axis
        zVal = accel.readAxis('z')-63;// Read Z Axis

        // Display them in the Serial Monitor window.
        Serial.print("X: = ");
        Serial.print(xVal, DEC);
        Serial.print("   Y: = ");
        Serial.print(yVal, DEC);
        Serial.print("   Z: = ");
        Serial.println(zVal, DEC);
        delay(1000);
```

```
  if (zVal > 40 || zVal < -40 || xVal >40 || xVal <-40 || yVal >
40 || yVal <-40){
    printLCD("HELP! I'm being","carried away!");
    delay(waitTime*4);
  }
```

4. Put your **noSteal()** function at the end of your **void loop()**. Test it - if you try to walk off with Barry he should complain.

5. Experiment with the values to see what happens when you make them a little bit different. Don't be frustrated if it is hard to work out 'what's going on' - the accelerometer can be complex to understand at first.