E11 - Autonomous Vehicles
Arduino Programming III
Given 4 unsigned binary numbers

0010, 0100, 1010, 1110

1. Convert the numbers to integers
2. Add the numbers, and output the sum of the numbers in the format:
   “The sum of a, b, c, d, is e”
3. Compare the sum with the number 20, and output the result in the format of an inequality:
   “The sum is (GREATER THAN) 20”
Outline

- Goal
- Review
- Modularity
- Multi-Dimensional Arrays
Quick Review

Functions & Function Calls

- A function definition, e.g.

  ```
  void sayHello()
  {
      Serial.println("Hello");
  }
  ```

- One or many calls to that function, e.g.

  ```
  void loop()
  {
      sayHello();
  }
  ```
Quick Review

- Conditionals
  - Using the if statement

```cpp
if (a > b) {
    Serial.println("Hello");
}
```
Quick Review

- **Using Arrays**
  - Lets store lists...
  - Set list element values
    
    ```
    listOfInts[2] = 201;
    ```
  
  - Get list element values
    
    ```
    int dummy = listOfInts[2];
    ```
For Loops

1. Let's declare an integer $i$ and set it to 0.
2. If $i$ is less than 10, then run all operations in the {} brackets that follow.
3. After running the operations, let $i=i+1$ and goto 2.

```c
for (int i=0; i<10; i++)
{
    speed = speed + 1;
}
```
Pop Quiz!

How do we use a for loop to count \( i \) down from 10 to 0? How many iterations will this do?

```java
for (int i=10; i>=0; i--) {
    speed = speed + i;
}
```
Quick Review

- Combine Loops and Arrays

```java
for (int i=0; i<10; i++)
{
    studentsGPA[i] = 0.4*i;
}
```
Quick Review

- Lets kick start our solution to today’s problem

```cpp
void setup()
{
  Serial.begin(9600);
}

void loop()
{
}
```
Outline

- Goal
- Review
- Multi-Dimensional Arrays
- Modularity
Multi-Dimensional Arrays

- One dimensional arrays stored a list of elements.
- E.g.
  
  ```cpp
  int list[5] = {2, 4, 9, 33 ,0};
  ```
Two-dimensional arrays store a list of lists of elements, (or matrix).

E.g.

```cpp
int mat[3][3] = {
    {2, 4, 9},
    {33, 0, 1},
    {-4, 55, 7}
};
```
Two-dimensional arrays store a list of lists of elements, (or matrix).

E.g.

```c
int mat[3][3] = {{2, 4, 9},
                 {33, 0, 1},
                 {-4, 55, 7}};
```

Row Index  Column Index
These are incredibly useful when combined with nested for loops

E.g.

```c
for (int i=0; i<3; i++){
    for (int j=0; j<3; j++){
        mat[i][j] = i+2*j;
    }
}
```
Multi-Dimensional Arrays

- Note we can pass around the $i^{\text{th}}$ list (or row) of the two-dimensional array.
- E.g.

  ```
  int mat[3][3];
  int list[3] = {1, 2, 3};
  mat[2] = list;
  ```
void setup()
{
    ... 
}

#define numBitsPerNumber 4
#define numNumbers 4
#define numBitsPerNumber 4
#define numNumbers 4

boolean binNumbers[numNumbers][numBitsPerNumber] =
{{0,0,1,0}, {0,1,0,0}, {1,0,1,0}, {1,1,1,0}};

void setup()
{
    ...
}
```c
#define numBitsPerNumber 4
#define numNumbers 4

boolean binNumbers[numNumbers][numBitsPerNumber] = {{0,0,1,0}, {0,1,0,0}, {1,0,1,0}, {1,1,1,0}};

The binary numbers we want to convert

void setup()
{
    ...
}
```
#define numBitsPerNumber 4
#define numNumbers 4

boolean binNumbers[numNumbers][numBitsPerNumber] = {{0,0,1,0}, {0,1,0,0}, {1,0,1,0}, {1,1,1,0}};

int intNumbers[numNumbers];

void setup()
{
    ...
}

Multi-Dimensional Arrays
#define numBitsPerNumber 4
#define numNumbers 4

boolean binNumbers[numNumbers][numBitsPerNumber] = 
{{0,0,1,0}, {0,1,0,0}, {1,0,1,0}, {1,1,1,0}};

int intNumbers[numNumbers];

void setup()
{
    ... 
}
Outline

- Goal
- Review
- Multi-Dimensional Arrays
- Modularity
Modularity

- Divide and Conquer
  - We will accomplish Modularity by breaking the problem into functions.

- Brainstorm
  - Let’s try to make a list of useful functions that may help us achieve our goal
Remember our goal...

Given 4 unsigned binary numbers

0010, 0100, 1010, 1110

1. Convert the numbers to integers
2. Add the numbers, and output the sum of the numbers in the format:
   “The sum of a, b, c, d, is e”
3. Compare the sum with the number 20, and output the result in the format of an inequality:
   “The sum is (GREATER THAN) 20”
Some functions (for us to write) that may help us achieve our goal

- convertToInteger()
- getSum()
- reportInequality()

- Lets use setup to call these functions
void setup()
{
    Serial.begin(9600);
}
void setup()
{
    Serial.begin(9600);

    for (int i=0; i<numNumbers; i++)
        intNumbers[i] = convertToInteger(binNumbers[i]);
}

Modularity
```cpp
void setup()
{
    Serial.begin(9600);

    for (int i=0; i<numNumbers; i++)
        intNumbers[i] = convertToInteger(binNumbers[i]);

    int sum = getSum(intNumbers);
}
```
void setup()
{
    Serial.begin(9600);

    for (int i=0; i<numNumbers; i++)
        intNumbers[i] = convertToInteger(binNumbers[i]);

    int sum = getSum(intNumbers);
    reportInequality(sum);
}

void setup()
{
    Serial.begin(9600);

    for (int i=0; i<numNumbers; i++)
        intNumbers[i] = convertToInteger(binNumbers[i]);

    int sum = getSum(intNumbers);
    reportInequality(sum);
}
Modularity

- convertToInteger()

```java
int convertToInteger(boolean bn[numBitsPerNumber]) {
}
```
int convertToInteger(boolean bn[numBitsPerNumber])
{
    int s = 0;

    return s;
}
```
int convertToInteger(boolean bn[numBitsPerNumber])
{
    int s = 0;
    int twoPowers[numBitsPerNumber] = {8,4,2,1};

    return s;
}
```
Modularity

- `convertTo Integer()`

```java
int convertToInteger(boolean bn[numBitsPerNumber])
{
    int s = 0;
    int twoPowers[numBitsPerNumber] = {8,4,2,1};

    for (int i=0; i<numBitsPerNumber; i++)
        s = s + bn[i]*twoPowers[i];

    return s;
}
```
Modularity

- `getSum()`

```c
int getSum(int numbers[numNumbers])
{
}
```
Modularity

- `getSum()`

```java
int getSum(int numbers[numNumbers])
{
    int s = 0;
    return s;
}
```
Modularity

- `getSum()`

```c
int getSum(int numbers[numNumbers])
{
    int s = 0;

    for (int i=0; i<numNumbers; i++)
    {

    }

    return s;
}
```
getSum()

```c
int getSum(int numbers[numNumbers])
{
    int s = 0;

    for (int i=0; i<numNumbers; i++)
    {
        s = s + numbers[i];
    }

    return s;
}
```
```cpp
int getSum(int numbers[numNumbers])
{
    int s = 0;
    Serial.print("The sum of ");
    for (int i=0; i<numNumbers; i++)
    {
        s = s + numbers[i];
        Serial.print(numbers[i]);
        Serial.print(" , ");
    }
    Serial.print("is ");
    Serial.println(s);
    return s;
}
```
Modularity

- reportInequality ()

```c
void reportInequality(int s) {
}
```
Modularity

- reportInequality()

```c
void reportInequality(int s) {
    Serial.print("The sum ");
    Serial.print(s);
    if (s>20)
        Serial.println(" is greater than 20.");
    else if (s<20)
        Serial.println(" is less than 20.");
    else
        Serial.println(" is equal to 20.");
}
```
Lab Reminder!

- Machine Shop Safety
  - Wear appropriate clothes
  - Hair tied back
  - Closed toed shoes
  - Goggles
  - Sleeves rolled up.
Lab Reminder!

- YOU MUST PASS THE SAFETY SHOP QUIZ!

And make sure to fill out AND SIGN the form...