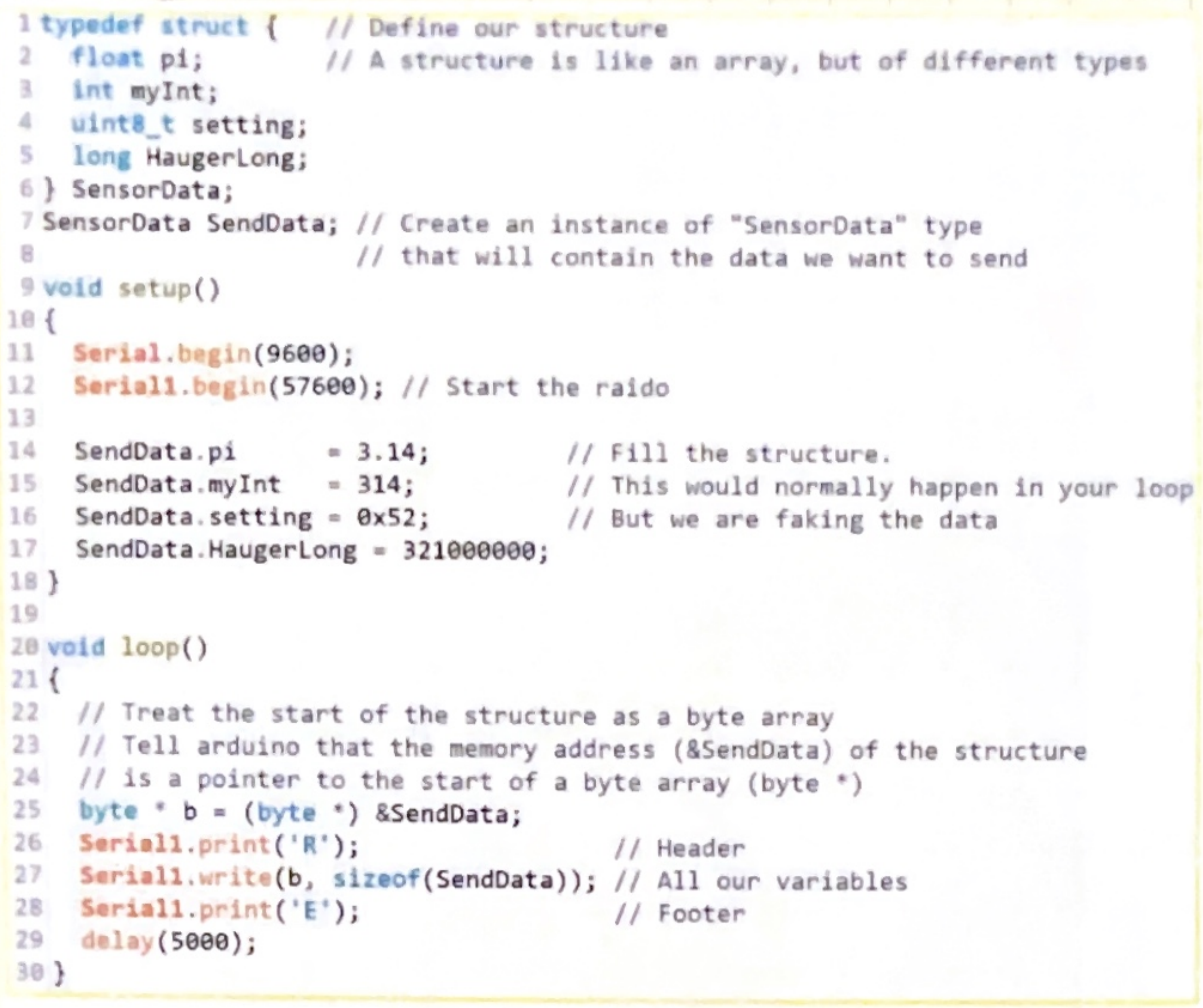
07/01/25
~~Wesley~~ Ean

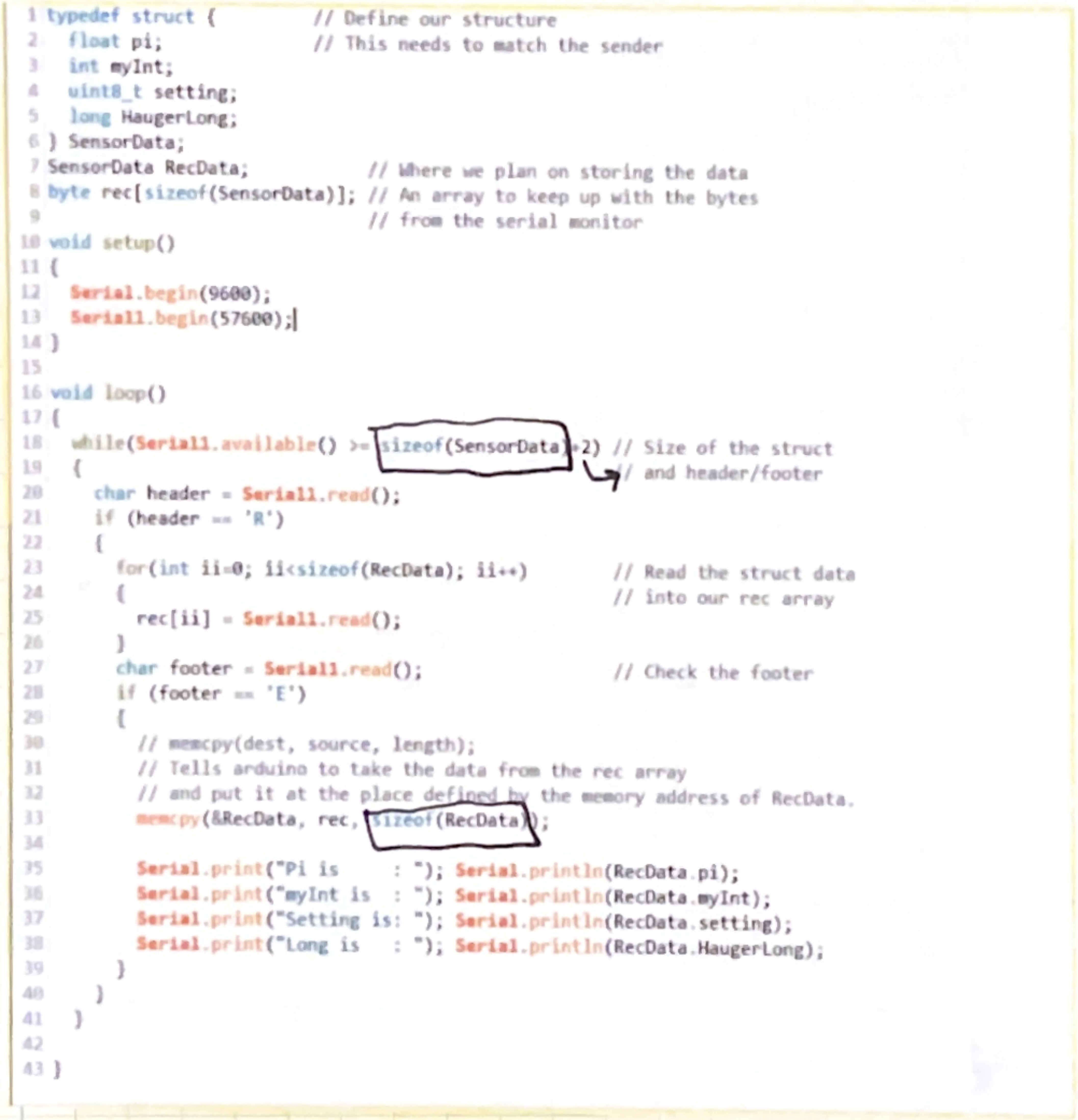
Serial Radio Advanced



```
typedef struct {    // Define our structure
  float pi;         // A structure is like an array, but of different types
  int myInt;
  uint8_t setting;
  long HaugerLong;
} SensorData;
SensorData SendData; // Create an instance of "SensorData" type
                     // that will contain the data we want to send
void setup()
{
  Serial.begin(9600);
  Serial1.begin(57600); // Start the raido

  SendData.pi      = 3.14;           // Fill the structure.
  SendData.myInt   = 314;            // This would normally happen in your loop
  SendData.setting = 0x52;           // But we are faking the data
  SendData.HaugerLong = 321000000;
}

void loop()
{
  // Treat the start of the structure as a byte array
  // Tell arduino that the memory address (&SendData) of the structure
  // is a pointer to the start of a byte array (byte *)
  byte * b = (byte *) &SendData;
  Serial1.print('R');                      // Header
  Serial1.write(b, sizeof(SendData)); // All our variables
  Serial1.print('E');                      // Footer
  delay(5000);
}
```



```
typedef struct {          // Define our structure
  float pi;               // This needs to match the sender
  int myInt;
  uint8_t setting;
  long HaugerLong;
} SensorData;
SensorData RecData;            // Where we plan on storing the data
byte rec[sizeof(SensorData)]; // An array to keep up with the bytes
                               // from the serial monitor
void setup()
{
  Serial.begin(9600);
  Serial1.begin(57600);
}

void loop()
{
  while(Serial1.available() >= sizeof(SensorData)+2) // Size of the struct
  {                                                   // and header/footer
    char header = Serial1.read();
    if (header == 'R')
    {
      for(int ii=0; ii<sizeof(RecData); ii++)        // Read the struct data
      {                                               // into our rec array
        rec[ii] = Serial1.read();
      }
      char footer = Serial1.read();                   // Check the footer
      if (footer == 'E')
      {
        // memcpy(dest, source, length);
        // Tells arduino to take the data from the rec array
        // and put it at the place defined by the memory address of RecData.
        memcpy(&RecData, rec, sizeof(RecData));

        Serial.print("Pi is      : "); Serial.println(RecData.pi);
        Serial.print("myInt is   : "); Serial.println(RecData.myInt);
        Serial.print("Setting is: "); Serial.println(RecData.setting);
        Serial.print("Long is    : "); Serial.println(RecData.HaugerLong);
      }
    }
  }

}
```

When sending a lot of data, using the ASCII text leaves variability in the number of bytes transmitted

For example:

"3.14" and "100.256" are

Both floats and have 4 bytes in memory. But the # of ASCII bytes are 4 and 7 respectively.

This makes it hard to know when to start receiving into memory.

If we use a struct and serial.write() instead, we know for certain how many bytes we are looking for.