Appendix B
Source Codes

1) `levelup_insulin_gen.mlx` - Dataset Generation

MATLAB

```
% Generation of dataset
x= randi([200,400],3000,1);
y_0=(.00009.*x.*x) + (0.2069.*x) + 22.847;
y=y_0+normrnd(4,0.3,3000,1);
dataset=[x y];



% Normalize the input features and target values
[input_x_norm, input_ps] = mapminmax(x', 0, 1);
input_x_norm = input_x_norm';
[y_0_norm, output_ps] = mapminmax(y', 0, 1);
y_0_norm = y_0_norm';



% Split data into training (80%) and testing (20%) sets
rng(42);     % to make the distribution between test and train fixed
cv = cvpartition(size(input_x_norm, 1), 'HoldOut', 0.2);
XTrain = input_x_norm(cv.training, :);
YTrain =dlarray( y_0_norm(cv.training, :));
XTest = input_x_norm(cv.test, :);
YTest = (y_0_norm(cv.test, :));


% Convert to dlarray format
XTrain = dlarray(XTrain);
XTest = dlarray(XTest);
```

2) `levelup_insulin_gen.mlx` - Neural Network Architecture and Training

MATLAB

```
% Building a neural network
net = dlnetwork;
layers = [
    featureInputLayer(1)
    fullyConnectedLayer(64)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(128)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(128)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(256)
    batchNormalizationLayer
    reluLayer
fullyConnectedLayer(256)
    batchNormalizationLayer
    reluLayer
    dropoutLayer(0.4)
fullyConnectedLayer(256)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(64)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(1)
    ];
net = addLayers(net, layers);

plot(net);
% analyzeNetwork(net);

  % defining the loss function - HUBER LOSS
function loss = dlhuber(y_pred, y_true, delta)
    e = y_pred - y_true;
    a = abs(e);
    mask = a <= delta;

    % Quadratic part (resembling L2)
    square_loss = 0.5 * e.^2;

    % Linear part (resembling L1)
    linear_loss = delta * (a - 0.5 * delta);

    % loss function equation
    loss = sum(mask .* square_loss + (~mask) .* linear_loss) /
numel(e);
end


% training the model
options = trainingOptions("adam", ...
    "Plots", "training-progress", ...
    "MaxEpochs", 200, ...
```

```matlab
    "InitialLearnRate", 0.001, ...
    "LearnRateSchedule", "piecewise", ...
    "LearnRateDropFactor", 0.1, ...
    "LearnRateDropPeriod", 50, ...
    "L2Regularization", 0.0005, ...
    "ValidationData", {XTest, YTest}, ...
    "ValidationFrequency", 90, ...
    "ValidationPatience", 10, ...
    "Shuffle", "every-epoch", ...
    "MiniBatchSize", 96);

huberLoss = @(y_pred, y_true) dlhuber(y_pred, y_true, 0.1);   % loss function

[net, info] = trainnet(XTrain, YTrain, net, huberLoss, options);

% predicting the values on the test data
y_pred_norm = predict(net, XTest);
% inverse to get the actual glucose values from the normalized data
y_pred = mapminmax('reverse', y_pred_norm.extractdata', output_ps)';
y_true = mapminmax('reverse', YTest', output_ps)';


% accuracy calculation
perc_error = abs((y_true - y_pred) ./ y_true) * 100;
accuracy_perc = 100 - mean(perc_error);


fprintf('Accuracy: %.2f%%\n', accuracy_perc);


% Plot actual vs predicted values
figure;
scatter(y_true, y_pred, 'filled');
hold on;
min_val = min([y_true; y_pred]);
max_val = max([y_true; y_pred]);
plot([min_val, max_val], [min_val, max_val], 'r--', 'LineWidth', 2);
hold off;
xlabel('Actual Glucose Levels');
ylabel('Predicted Glucose Levels');
title('Actual v/s Predicted Glucose Levels');
legend('Predictions','Perfect Prediction Line');
grid on;
```

3) `tf_to_tflite.py` - Converting TensorFlow model to TFLite model

Python – VS Code

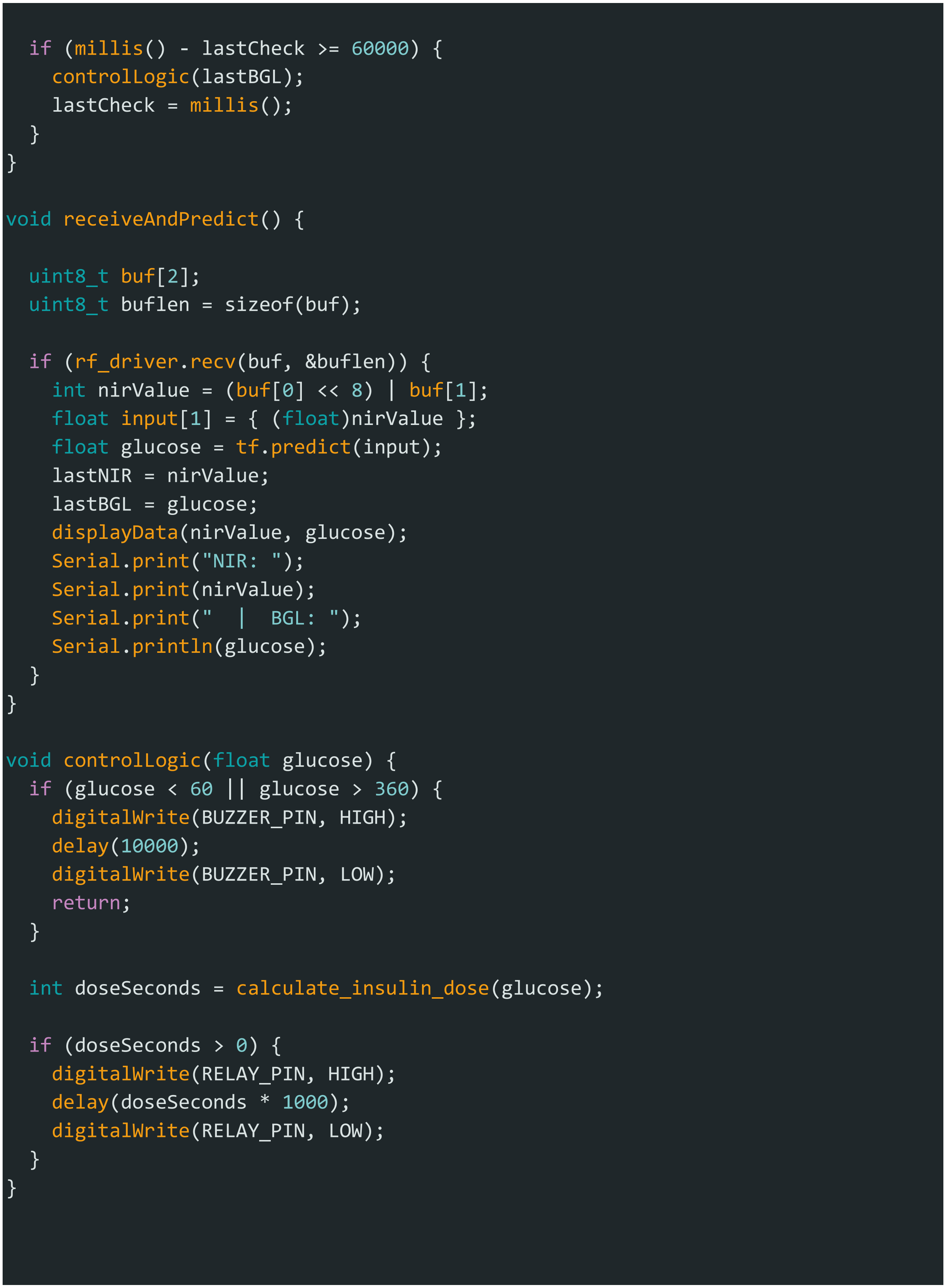
```python
import tensorflow as tf

# Load SavedModel
saved_model_dir = "nir_to_glucose"
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)

# Optimize for size (Quantization)
converter.optimizations = [tf.lite.Optimize.DEFAULT]

# Convert to TFLite
tflite_model = converter.convert()

# Save TFLite Model
with open("nir_to_glucose.tflite", "wb") as f:
    f.write(tflite_model)

print("TFLite model conversion successful! File saved as 
'nir_to_glucose.tflite'")
```

4) `insulin_pump.ino` - Mechanism for the Insulin Pump (controlled by ESP32)

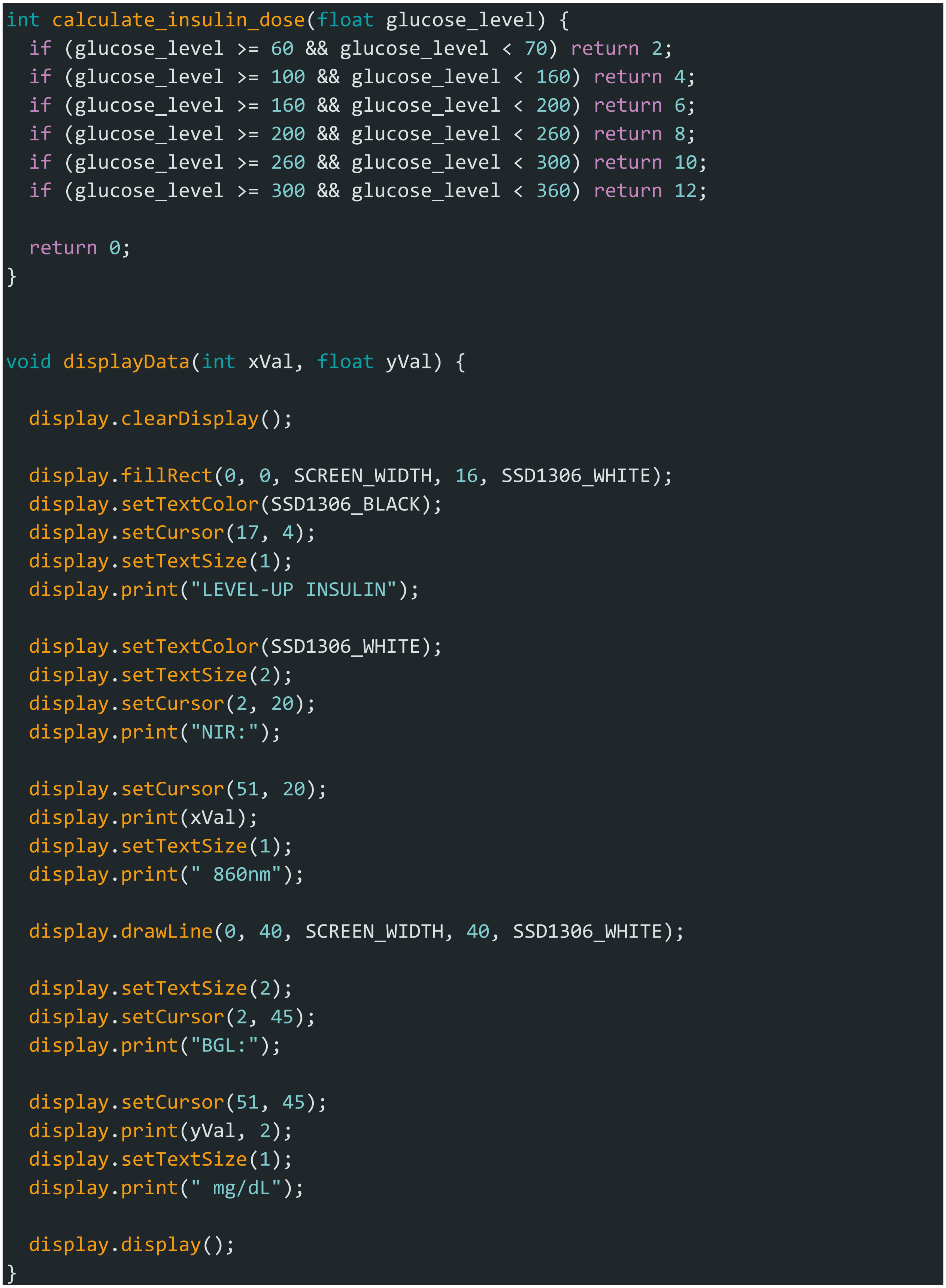
Arduino IDE

```cpp
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <RH_ASK.h>
#include <SPI.h>
#include "nir_to_glucose.h"
#include <tflm_esp32.h>
#include <eloquent_tinyml.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

#define RELAY_PIN 27
#define BUZZER_PIN 14
```

```
#define NUM_OF_INPUTS 1
#define NUM_OF_OUTPUTS 1
#define ARENA_SIZE 2 * 1024

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

RH_ASK rf_driver;

Eloquent::TinyML::TfLite<NUM_OF_INPUTS, NUM_OF_OUTPUTS, ARENA_SIZE>

tf(nir_to_glucose);

unsigned long lastCheck = 0;
float lastBGL = 0;
int lastNIR = 0;

void setup() {

  Serial.begin(115200);

  pinMode(RELAY_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);

  tf.begin();

  if (!rf_driver.init()) {
    Serial.println("RF init failed");
    while (true);
  }

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println("SSD1306 failed");
    while (true);
  }

  display.clearDisplay();
  display.display();
}

void loop() {

  receiveAndPredict();
```

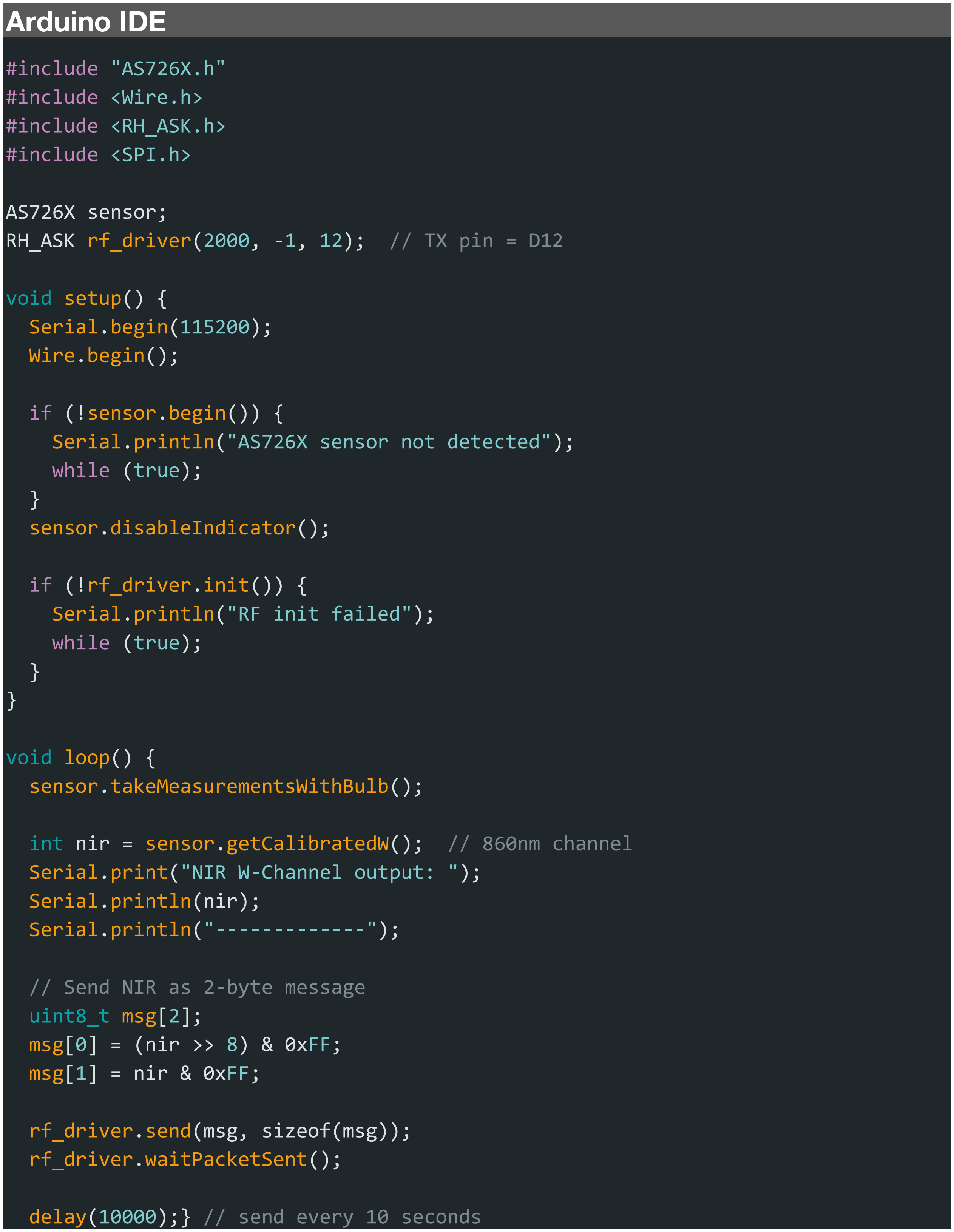


```
  if (millis() - lastCheck >= 60000) {
    controlLogic(lastBGL);
    lastCheck = millis();
  }
}

void receiveAndPredict() {

  uint8_t buf[2];
  uint8_t buflen = sizeof(buf);

  if (rf_driver.recv(buf, &buflen)) {
    int nirValue = (buf[0] << 8) | buf[1];
    float input[1] = { (float)nirValue };
    float glucose = tf.predict(input);
    lastNIR = nirValue;
    lastBGL = glucose;
    displayData(nirValue, glucose);
    Serial.print("NIR: ");
    Serial.print(nirValue);
    Serial.print("  |  BGL: ");
    Serial.println(glucose);
  }
}

void controlLogic(float glucose) {
  if (glucose < 60 || glucose > 360) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(10000);
    digitalWrite(BUZZER_PIN, LOW);
    return;
  }

  int doseSeconds = calculate_insulin_dose(glucose);

  if (doseSeconds > 0) {
    digitalWrite(RELAY_PIN, HIGH);
    delay(doseSeconds * 1000);
    digitalWrite(RELAY_PIN, LOW);
  }
}
```



```
int calculate_insulin_dose(float glucose_level) {
  if (glucose_level >= 60 && glucose_level < 70) return 2;
  if (glucose_level >= 100 && glucose_level < 160) return 4;
  if (glucose_level >= 160 && glucose_level < 200) return 6;
  if (glucose_level >= 200 && glucose_level < 260) return 8;
  if (glucose_level >= 260 && glucose_level < 300) return 10;
  if (glucose_level >= 300 && glucose_level < 360) return 12;

  return 0;
}


void displayData(int xVal, float yVal) {

  display.clearDisplay();

  display.fillRect(0, 0, SCREEN_WIDTH, 16, SSD1306_WHITE);
  display.setTextColor(SSD1306_BLACK);
  display.setCursor(17, 4);
  display.setTextSize(1);
  display.print("LEVEL-UP INSULIN");

  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(2);
  display.setCursor(2, 20);
  display.print("NIR:");

  display.setCursor(51, 20);
  display.print(xVal);
  display.setTextSize(1);
  display.print(" 860nm");

  display.drawLine(0, 40, SCREEN_WIDTH, 40, SSD1306_WHITE);

  display.setTextSize(2);
  display.setCursor(2, 45);
  display.print("BGL:");

  display.setCursor(51, 45);
  display.print(yVal, 2);
  display.setTextSize(1);
  display.print(" mg/dL");

  display.display();
}
```

4) `CGM_sensor.ino -` Mechanism for the CGM sensor (controlled by Arduino Nano)



Arduino IDE

```
#include "AS726X.h"
#include <Wire.h>
#include <RH_ASK.h>
#include <SPI.h>

AS726X sensor;
RH_ASK rf_driver(2000, -1, 12);  // TX pin = D12

void setup() {
  Serial.begin(115200);
  Wire.begin();

  if (!sensor.begin()) {
    Serial.println("AS726X sensor not detected");
    while (true);
  }
  sensor.disableIndicator();

  if (!rf_driver.init()) {
    Serial.println("RF init failed");
    while (true);
  }
}

void loop() {
  sensor.takeMeasurementsWithBulb();

  int nir = sensor.getCalibratedW();  // 860nm channel
  Serial.print("NIR W-Channel output: ");
  Serial.println(nir);
  Serial.println("-------------");

  // Send NIR as 2-byte message
  uint8_t msg[2];
  msg[0] = (nir >> 8) & 0xFF;
  msg[1] = nir & 0xFF;

  rf_driver.send(msg, sizeof(msg));
  rf_driver.waitPacketSent();

  delay(10000);} // send every 10 seconds
```