COE306

Embedded Systems Smart Retail Platform

by

Talal Khaled Al-Harbi

202036240-306-51

for

Dr. Hazem Selmi

Intro. To Embedded Systems

Descriptive Abstract

This report describes the development of a Smart Load Monitoring System using an ESP32 microcontroller, HX711 ADC, and load sensors, to be used in retail stores for the purpose of automating purchasing experience. The project aims to accurately measure weights and count items placed on a platform by leveraging the HX711 module and the Wi-Fi connectivity features of the ESP32. The report covers the hardware and software architecture, component design, system integration, and testing, along with a discussion on the results and challenges faced during the project.

18 May 2024

INTRODUCTION

The problem addressed by this project is the need for a precise and connected load monitoring system capable of measuring weights and counting items. The system should be able to communicate data wirelessly, allowing for remote monitoring and control. The requirements include accurate weight measurement, wireless connectivity, and a user-friendly interface for the user of this platform. These requirements are translated into specifications such as the use of an ESP32 microcontroller for its WiFi capabilities, HX711 ADC for precise weight measurement, and Arduino Cloud for data management and remote session access.

ARCHITECTURE

Hardware Architecture

- **ESP32 Microcontroller**: Chosen for its built-in WiFi module, the ESP32 serves as the central unit of the system, handling data acquisition, processing, and communication.
- **HX711 ADC**: A precision 24-bit ADC used to convert the analog signals from the load sensors to digital data. The HX711 operates over the I2C protocol.
- **Load Sensors**: Four degree load sensors mounted on a wooden platform via a 3D printed module. These sensors provide the raw weight data to the HX711.

Software Architecture

- **HX711_ADC Library**: Utilized to interface with the HX711 module, allowing for easy data reading and calibration.
- **Arduino Cloud**: Used for remote cloud variables, visualization, and session control.
- **Custom Code**: Developed for data processing, including weight measurement, item counting, balance calculation, and improper purchasing/laoding of items when a session has not started.

DESIGN AND DESIGN ALTERNATIVES

1. **Initial Design**: Large and thin wooden platform. Failed due to inaccurate readings and inaccurate force distribution that affected the output data from the HX711 module.
2. **Final Design**: Smaller and thicker wooden platform. Provided more accurate readings with reduced error.

The final design was selected based on improved accuracy and stability in readings.

WHAT I LEARNED

- **Load Sensor Output**: The output voltage of the load sensors is between 0 to 50 millivolts, that is why we needed the HX711 amplifier module with selectable gain levels of 32, 64, and 128.
- **Item Counting**: The round function was found to be the best method for calculating the number of items due to the high error potential in sensor readings.

PROBLEMS FACED

- **Platform Instability**: The initial thin platform led to inaccurate readings.
- **Sensor Calibration**: Ensuring accurate calibration of the load sensors was challenging, which takes about 5-7 minutes to almost stablize readings.
- **Wireless Connectivity**: Connecting to the KFUPM Wi-Fi and personal cellilure network failed to connect the ESP32 to the cloud. However using a home Wi-Fi did the trick.

RESULT AND DISCUSSION

The final design provided accurate count, and not weight, measurements with minimal error, meeting the project goals. The use of a smaller, thicker platform significantly improved the system's performance. The integration with Arduino Cloud allowed for effective remote monitoring, making the system user-friendly and efficient.

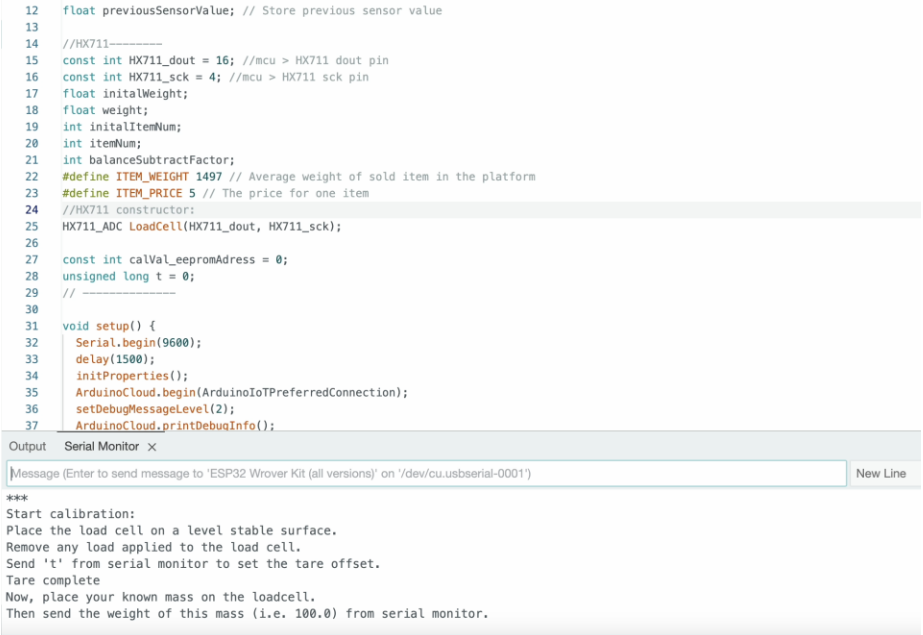
IMPORTANT FUNCTIONS DESCRIPTION

ArduinoCloud.update(): **Used in every loop to get updates from the cloud, otherwise we won't be able to recive changes on our cloud reading variables.**

LoadCell.update(): **Detectes when the HX711 module sends new data serially.**

round(weight / ITEM_WEIGHT): **Used the round function to calculate the item number**

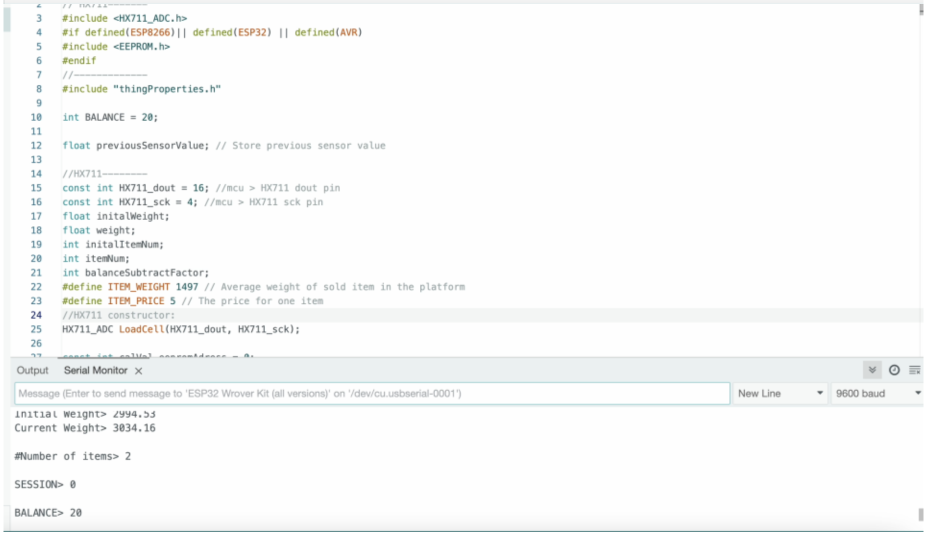
SAMPLE RUNS



```
float previousSensorValue; // Store previous sensor value

//HX711--------
const int HX711_dout = 16; //mcu > HX711 dout pin
const int HX711_sck = 4; //mcu > HX711 sck pin
float initalWeight;
float weight;
int initalItemNum;
int itemNum;
int balanceSubtractFactor;
#define ITEM_WEIGHT 1497 // Average weight of sold item in the platform
#define ITEM_PRICE 5 // The price for one item
//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;
// --------------

void setup() {
  Serial.begin(9600);
  delay(1500);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
```

Output
Serial Monitor
Message (Enter to send message to 'ESP32 Wrover Kit (all versions)' on '/dev/cu.usbserial-0001')
New Line

```
***
Start calibration:
Place the load cell on a level stable surface.
Remove any load applied to the load cell.
Send 't' from serial monitor to set the tare offset.
Tare complete
Now, place your known mass on the loadcell.
Then send the weight of this mass (i.e. 100.0) from serial monitor.
```



```
// HX711--------
#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif
//-------------
#include "thingProperties.h"

int BALANCE = 20;

float previousSensorValue; // Store previous sensor value

//HX711--------
const int HX711_dout = 16; //mcu > HX711 dout pin
const int HX711_sck = 4; //mcu > HX711 sck pin
float initalWeight;
float weight;
int initalItemNum;
int itemNum;
int balanceSubtractFactor;
#define ITEM_WEIGHT 1497 // Average weight of sold item in the platform
#define ITEM_PRICE 5 // The price for one item
//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
```

Output
Serial Monitor
Message (Enter to send message to 'ESP32 Wrover Kit (all versions)' on '/dev/cu.usbserial-0001')
New Line
9600 baud

```
Initial Weight> 2994.53
Current Weight> 3034.16

#Number of items> 2

SESSION> 0

BALANCE> 20
```

CONCLUSION

The Smart Load Monitoring System successfully achieved its goals of accurate weight measurement and item counting with wireless connectivity. The project highlighted the importance of sensor calibration and platform stability. Future improvements could include exploring different materials for the platform and enhancing the user interface for easier calibration and monitoring.