Smart Ballot Box (SBB) 2019 - PSU Edition

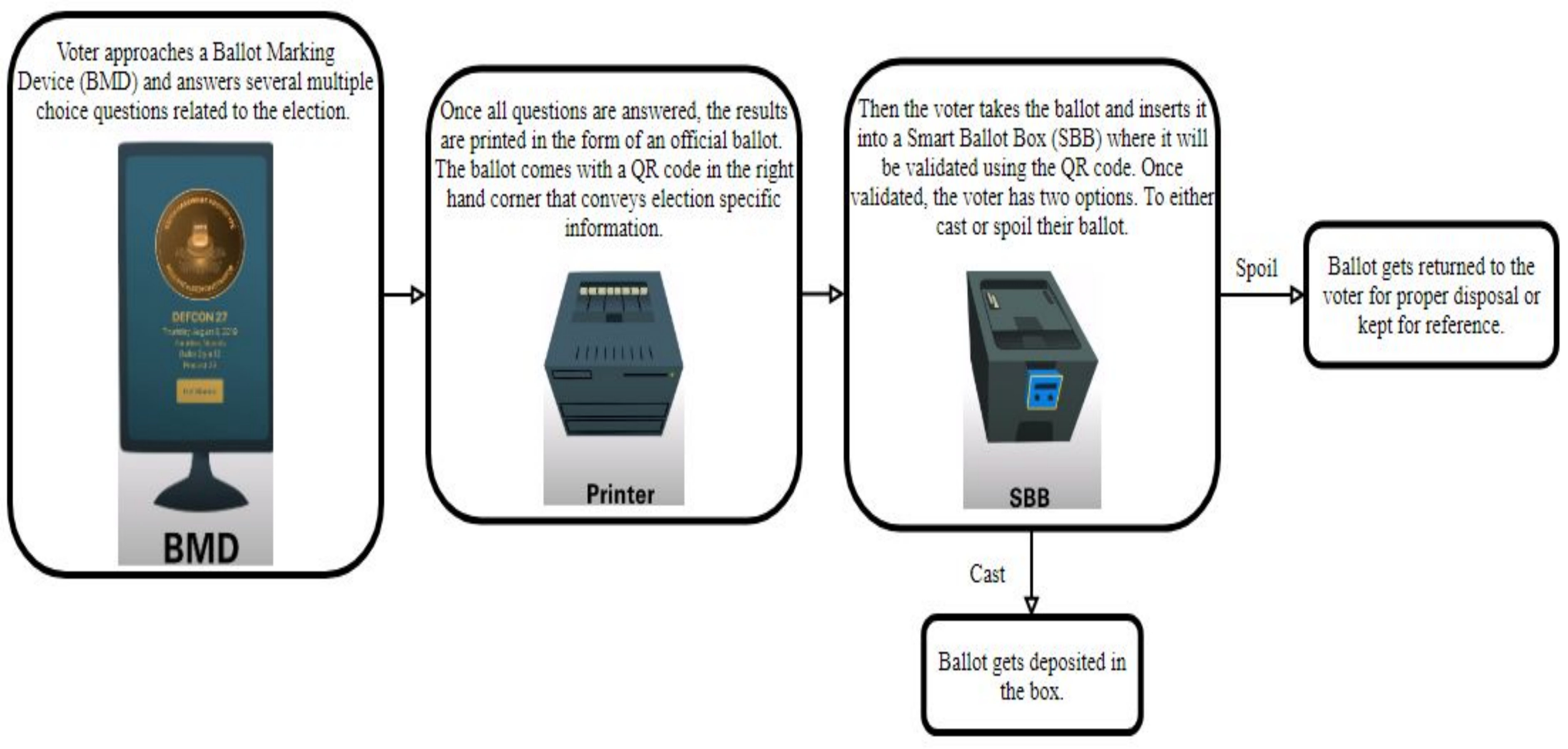
Team 4 -
Ali Saad, Jonathan Christian,
Nicholas Long and Jiaqi Liu
Faculty Advisor: Tom Schubert
Sponsor: Free & Fair

Project Background

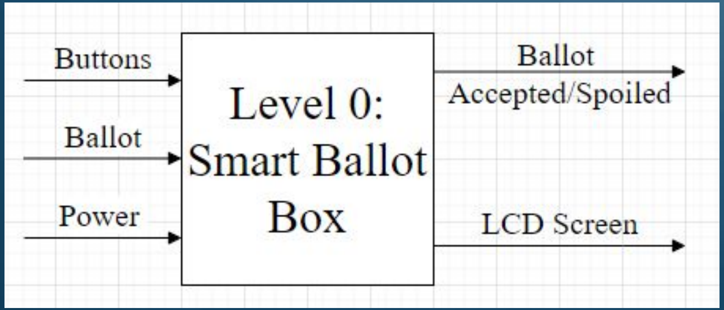
- Problem or Need: FPGA CPU over $9,000
- Motivation: Find a full prototype, affordable
- Objective: Port existing software and functionality to consumer friendly platform
- Alternatives: SBB(2019), $9,000 FPGA
- Requirements: Maintain the functionality and make it more cost effective.
- Our Approach : Use Arduino Uno and CASCADIO Board instead of FPGA, and code using FreeRTOS
- Reduced scope and deliverables

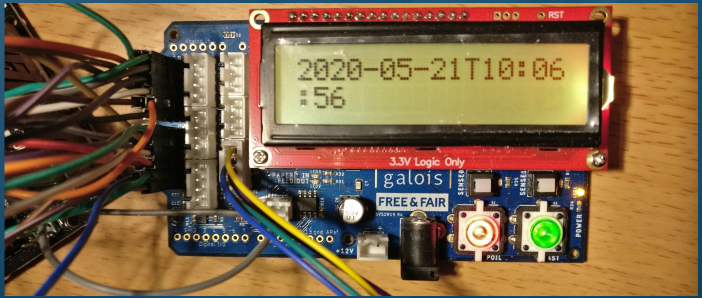
Needs	Objectives
Most voting centers cannot afford to spend $9000 + for a single ballot box	Make an affordable smart ballot box
The current prototype can't be easily replicated	Make the smart ballot box easy to replicate
The current COTS components used are not available everywhere	Use widely available COTS parts for the smart ballot box
May not be safe to produce without providing training	Make the smart ballot box safe to operate and manufacture
The current prototype uses some custom parts, not easily obtained outside of the manufacturer	Limit the number of custom parts and favor COTS components
The cost prevents private citizens from exploring the established security features	Promote experimentation of the smart ballot box to private citizens and tinkers

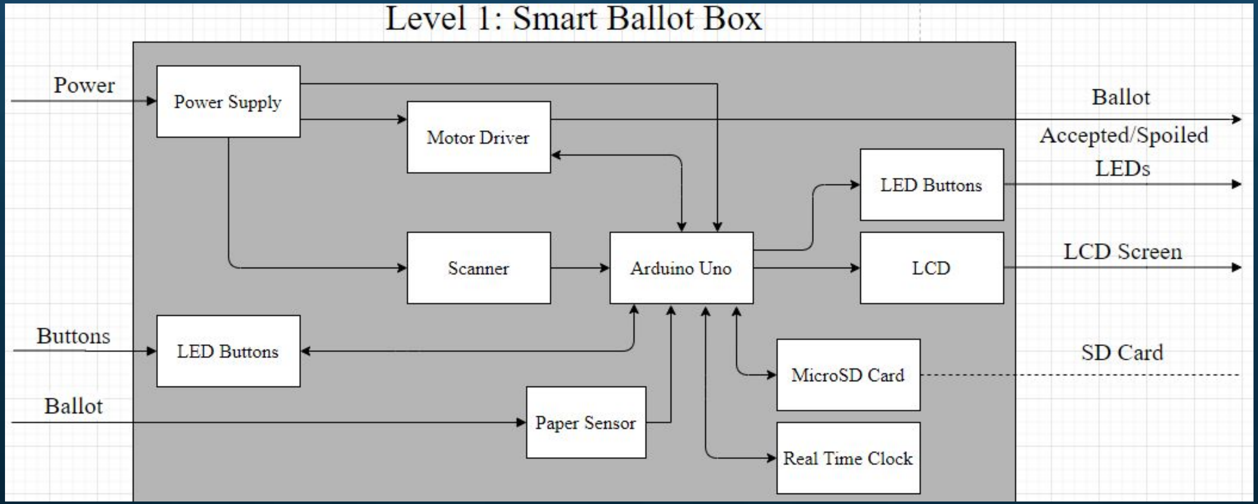
Voter Flow Chart



Architectural Overview / Block Diagrams



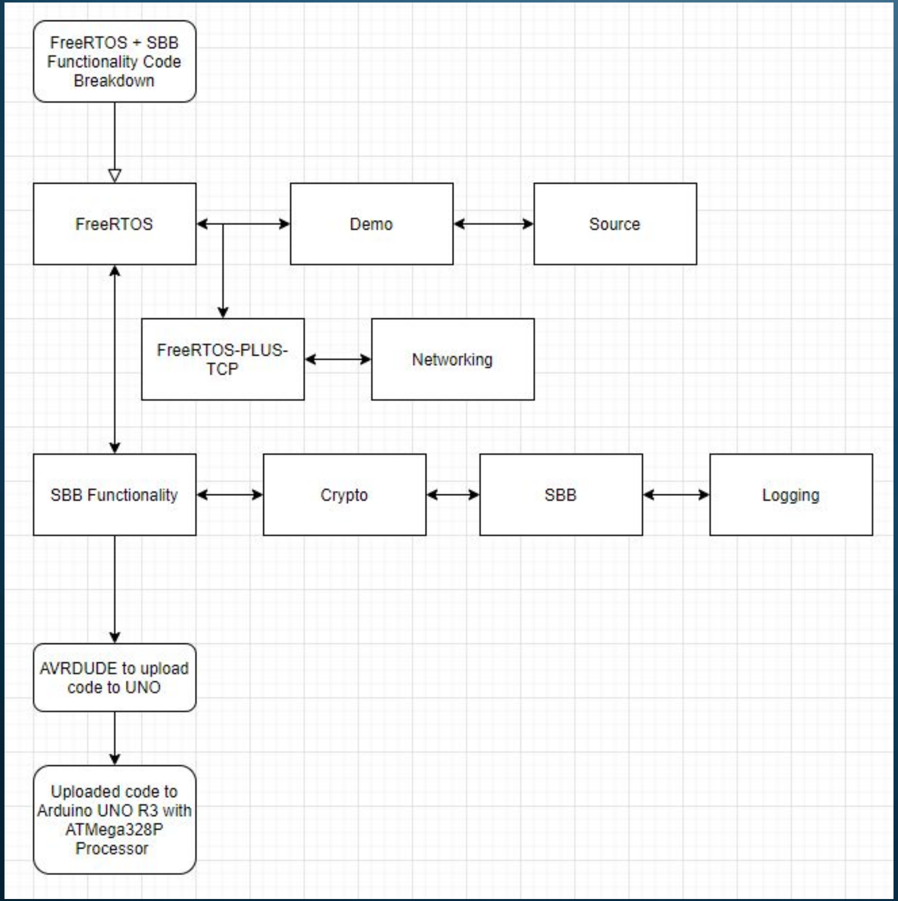


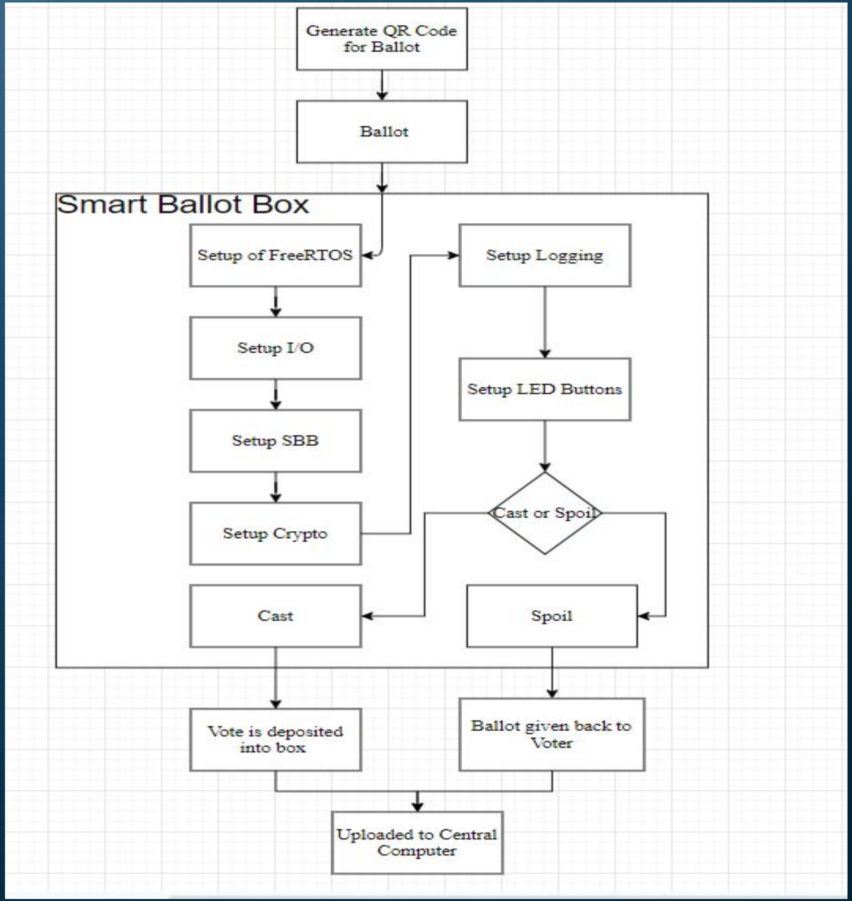


Software/Code

- Started with the Arduino IDE to upload FreeRTOS
- Switched to command line tools and Ubuntu
- FreeRTOS published code for Atmega323 but not ATmega328P
- Injected the SBB code in our filesystem
- Added a switch to the makefile to compile for ATmega328P
- Used AVRDUDE and AVR-GCC

Software Diagram





Results

- UNO platform has insufficient memory for full SBB functionality
- SBB functionality optimized for RISC processor
- FreeRTOS +SBB consumes nearly 370 KBits of memory
- Would need to select an upgraded microcontroller for full prototype

File Size				
	Free & Fair Results (Prior to optimization)	Free & Fair Results (Optimized)	Team 4 Results (Our FreeRTOS with Free & Fair SBB code)	FreeRTOS running with Blinking LED (before importing with Free & Fair Code)
FreeRTOS Results:	188k	122k	72K	11k
SBB Results:	178k	115k	112k	N/A
Crypto Results (Part of SBB Code):	66k	N/A	11.7k	N/A
Networking	N/A	N/A	N/A	N/A
Logging	N/A	N/A	N/A	N/A
Total Memory Result:	366k	237k	184k	N/A

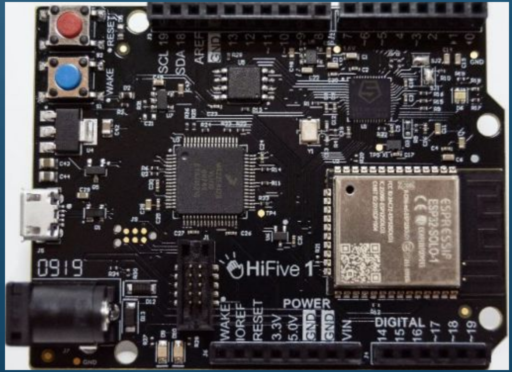
Possible Boards based on Results

Arduino Due $40

32-Bit ARM Core Microprocessor

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz

HiFive1 Rev B $60



32-Bit SiFive E31 RISC-V Core

Input Voltage	5 V USB or 7-12 VDC Jack
IO Voltage	3.3 V
Digital I/O Pins	19
PWM Pins	9
External Wakeup Pins	1
Flash Memory	32 Mbit Off-Chip (ISSI SPI Flash)

Key Takeaways

- Techniques to continue development of code while keeping original functionality.
- Improved our coding skills overall. Learned a great deal about Make and using multiple makefiles.
- Practiced how to communicate at a professional yet technical level with fellow engineers.
- Testing procedures and debugging
- System analysis routine through schematic and diagram review of an unfamiliar project
- Ways of reducing security risks though crypto and functional operations

Thank You!

- Joe Kiniry
- Daniel Zimmerman
- Steven Osborn
- Joey Dodds
- Tom Schubert
- Andrew Greenberg
- Mark Faust

Questions or Comments?

References

- Arduino, 2020. "Arduino Due." Arduino Due | Arduino Official Store, store.arduino.cc/usa/due.
- SiFive,Inc. "SiFive." Https://www.sifive.com/Share.png, www.sifive.com/boards/hifive1-rev-b.
- Free & Fair animation of voter system