System Driven Hardware Design (SDHD)

Movement Detection Implementation

Author: Aadarsh Kumar Singh,

Matrikel Nummer 766499, Embedded Systems and Microelectronics

Contents

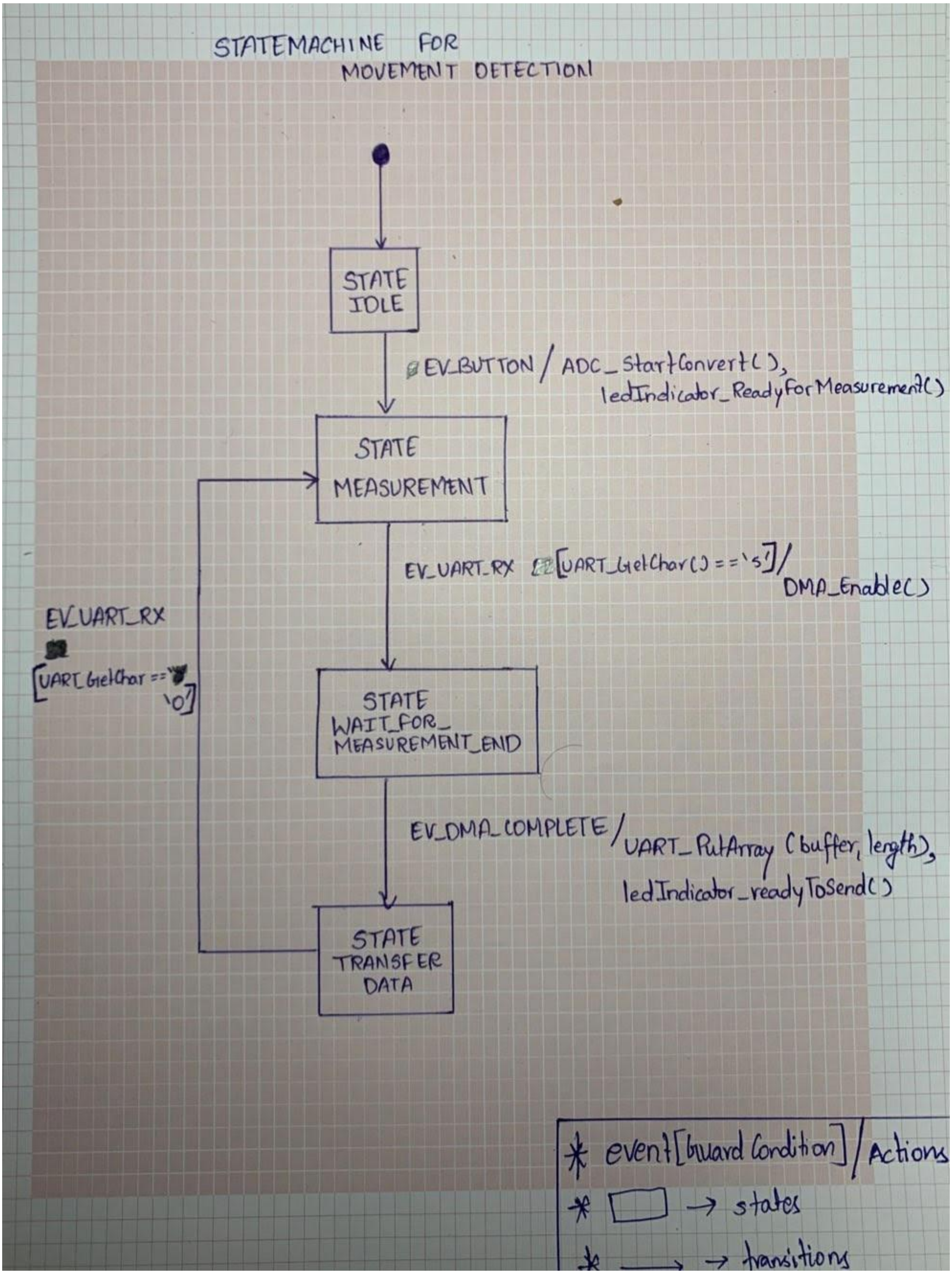
1. State Machine 2

2. Picture of the Hardware Setup 4

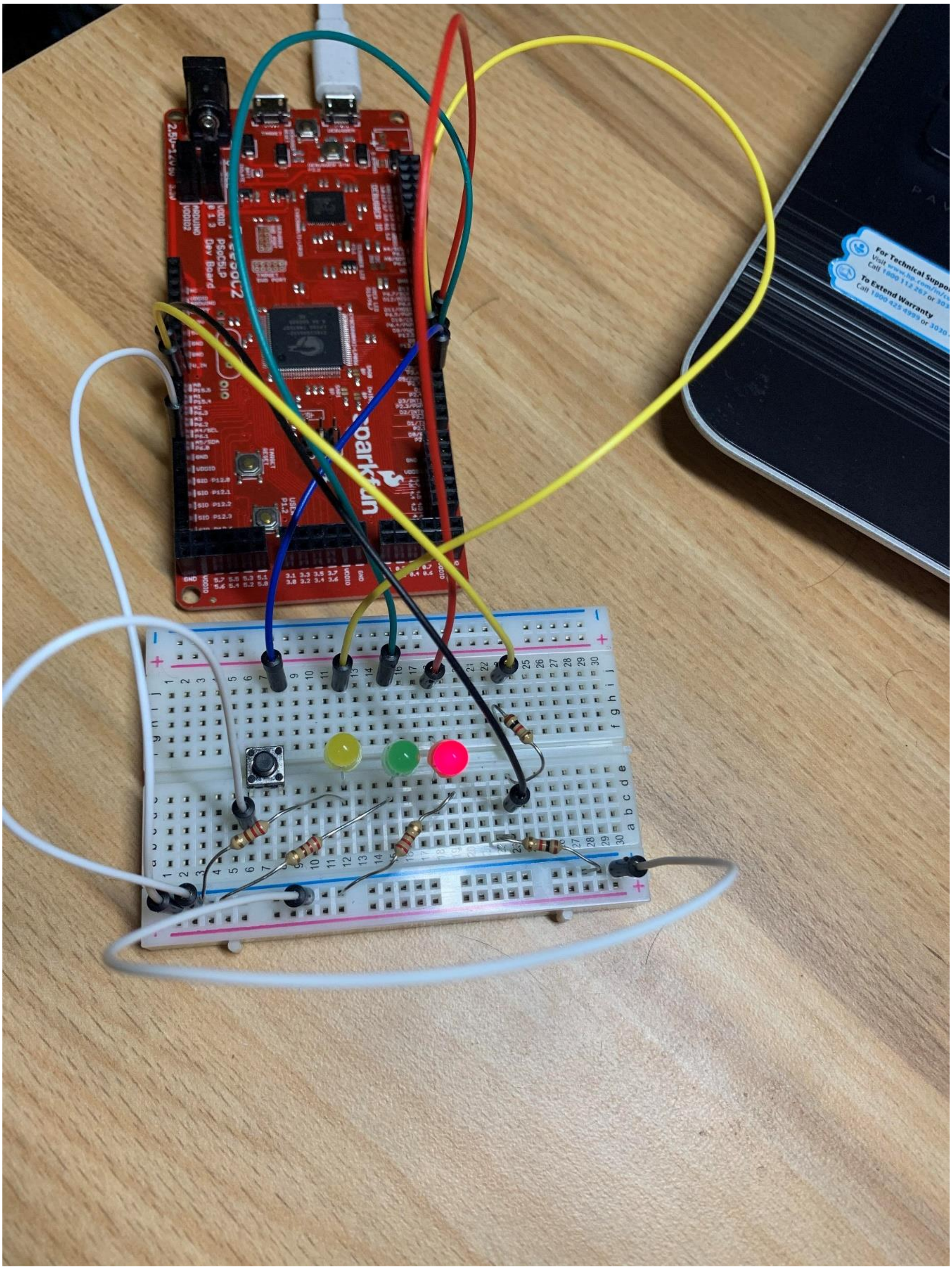
3. Ideas for Improvement and Design Decisions: 5

1. State Machine

- The system has 4 states:
  - STATE_IDLE
  - STATE_MEASUREMENT
  - STATE_STATE_WAIT_FOR_MEASUREMENT_END
  - STATE_TRANSFER_DATA
- The system has 3 events:
  - EV_BUTTON: A Flag is set when a button is pressed
  - EV_UART_RX: Another Flag is set when a byte is received in UART
  - EV_DMA_COMPLETE: Another Flag is set When DMA read is completed
- It has two guard conditions for the event (EV_UART_RX):
  - The byte received is ‘s’ – Start condition for the transfer
  - The byte received is ‘o’ – End condition when ADC samples (2048 bytes) are transferred.
- The diagram of the state machine is attached below:


STATEMACHINE FOR MOVEMENT DETECTION
STATE IDLE
EV_BUTTON / ADC_StartConvert( ), ledIndicator_ReadyForMeasurement( )
STATE MEASUREMENT
EV_UART_RX [UART_GetChar( ) == 's'] / DMA_Enable( )
STATE WAIT_FOR_ MEASUREMENT_END
EV_DMA_COMPLETE / UART_PutArray (buffer, length), ledIndicator_readyToSend( )
STATE TRANSFER DATA
EV_UART_RX [UART_GetChar == 'o']
* event[Guard Condition]/Actions
* → states
* → transitions

2. Picture of the Hardware Setup



3. Ideas for Improvement and Design Decisions:

- The state machine is designed in such a way that the receiver (MATLAB) always receives the latest ADC samples, i.e. if the user has pressed button to start the ADC sampling and then MATLAB script is started even then the user will get latest ADC samples.
  Hence it doesn't matter if the user presses the button first or if the receiver is started first, In both the situations, receiver will receive latest ADC samples.

- Ideas for improvement:
  - RTOS Can be used, It will improve the timing behavior and can be very useful to give priorities to the tasks (function called by the scheduler), higher priorities task will be called first then the lower priority tasks.
  - RTOS can handle the race condition better, Imagine the scenario when The user presses the button and at the same time there is a byte received at UART , in this case two events occur at the same time, Although this is a rare chance but this might happen, Hence bringing in RTOS can solve this race conditions.
  - Events In RTOS are cleared as soon as it is processed whereas in bare metal implementation like ours, the flags need to be carefully reset after processing.

  Although, RTOS increases the complexity of the system, but is highly useful for managing the timing behavior, setting priorities and event management of the tasks(function called by the scheduler).