Stanford ME218B:
Smart Product Design Lab

Hi-tech Indoor Ping-Pong Orbital Sumo

Pseudo code
2012.03.12

Team 3

Alves, Dimitri

Anderson, Ryan

Ploch, Chris

Yang, Yushi

Master State Machine: RobotMotionSM

InitRobotMotionSM

Assign ThisEvent to ES_Event
Save our priority
Assign ThisEvent Event type to ES_ENTRY
Start the Master State machine
Return true
End InitRobotMotionSM

PostRobotMotionSM

Return MyPriority and ThisEvent
End PostRobotMotionSM

RunRobotMotionSM

Assume no transitions are made
Assign Next State to Current State
Default to normal entry to new state
Assume we are not consuming event

Based on current state, switch through the following cases

Case: if current state is WaitingState
If an event is active
Switch based on current event type
Case: Beacon 1 is detected we are team BLUE
Light Blue LED
Consume this event
Break of case Beacon 1 is detected

Case: Beacon 3 is detected we are team RED
Light Red LED
Consume this event
Break of case Beacon 3 is detected

Case: event is Game On from SPI
Initialize one timer for total game on time (2 min)
Initialize timer for total ball collecting time
Start rotating the robot clockwise
Initialize timer to record rotating to center time
Next State is BallCollectingState
Mark that we are taking a transition

Consume this event
Break of Game On event case

End switch cases statement
End if statement
Break of WaitingState

Case: if current state is BallCollectingState
Call the during function to run RunBallCollectionSM
If an event is active
Switch based on current event type
Case: Timeout
if event is timeout from ROBOT_MOTION
Start rotating the robot clockwise to look for beacon
Next state is BallDumpingState
Mark that we are taking a transition
Break of timeout case

End switch cases statement
End if statement
Break of BallCollectingState

Case: if current state is BallDumpingState
Call the during function to run RunBallDumpSM
If an event is active
Switch based on current event type
Case: If event is FinishDumping
Next state is DefendingState
Mark that we are taking a transition
Break of FinishDumping case

Case: Timeout
if event is timeout for 2 min game over
Stop the robot
Turn on both LEDs
Next state is WaitingState
Mark that we are taking a transition
Break of timeout case
End switch cases statement
End if statement
Break of BallDumpingState

Case: if current state is BallDefendingState
Call the during function to run RunBallDefenseSM
If an event is active
Switch based on current event type
Case: Timeout
if event is timeout for 2 min game over
Stop the robot
Turn on both LEDs
Next state is WaitingState
Mark that we are taking a transition
Break of timeout case
End switch cases statement
End if statement
Break of BallDefendingState
End of switch states

If we are making a transition
Execute exit function for current state
Modify state variable
Execute entry function for new state
End if statement

Return ReturnEvent

End of RunRobotMotionSM

StartRobotMotionSM
Always start at WaitingState
call the entry function (if any) for the ENTRY_STATE
return nothing
end of StartRobotMotionSM

QueryRobotMotionSM
Return Current State of the RobotMotionSM
End of QueryRobotMotionSM

DuringBallCollectingState
If event is ES_ENTRY
Start BallCollectionSM
Else if event is ES_EXIT

give the lower levels a chance to clean up first
Else
Do the 'during' function for this state: run RunBallCollectionSM
End of DuringBallCollectingState

DuringBallDumpingState
If event is ES_ENTRY
Start BallDumpSM
Else if event is ES_EXIT
give the lower levels a chance to clean up first
Else
Do the 'during' function for this state: run RunBallDumpSM
End of DuringBallDumpingState

DuringBallDefendingState
If event is ES_ENTRY
Start BallDefenseSM
Else if event is ES_EXIT
give the lower levels a chance to clean up first
Exit function for BallDefendingState: Stop the robot
Else
Do the 'during' function for this state: run RunBallDefenseSM
End of DuringBallDefendingState

Sub-level State Machine 1: BallCollectionSM

RunBallCollectionSM

Assume no transition made
Assign NextState to CurrentState
Assume we are not consuming event

Based on current state, switch through the following cases

Case: If current state is RotatingToCenter
If an event is active
Switch based on current event type
Case: timeout event
If timeout is ball collection timer, then robot has rotate to center
Start driving the robot forward
Reinitialize timer to 'forward to center' time
Next State will be DrivingToCenter State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout
End of if event is active statement
Break of RotatingToCenter

Case: If current state is DrivingToCenter
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has driven to center
Start rotating robot clockwise
Reinitialize timer to '90 degrees rotation' time
Next State will be RotatingCWToForward1 State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

Case: LeftFrontBumpSensor triggered event
Start rotating robot clockwise
Reinitialize timer to '90 degrees rotation' time
Next State will be RotatingCWToForward1 State
Mark that we are making a transition

Consume this event
Break of case LeftRearBumpSensor

Case: RightFrontBumpSensor triggered event
Start rotating robot clockwise
Reinitialize timer to '90 degrees rotation' time
Next State will be RotatingCWToForward1 State
Mark that we are making a transition
Consume this event
Break of case RightRearBumpSensor

End of switch event type
End of if event is active statement
Break of DrivingToCenter

Case: If current state is RotatingCWToForward1
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot is parallel to the wall
Start driving the robot forward
Reinitialize timer to 'center to wall' time
Next State will be DrivingForwardToWall State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

Case: LeftRearBumpSensor triggered event
Start driving the robot forward
Reinitialize timer to ' center to wall ' time
Next State will be DrivingForwardToWall State
Mark that we are making a transition
Consume this event
Break of case LeftRearBumpSensor

Case: RightRearBumpSensor triggered event
Start driving the robot forward
Reinitialize timer to ' center to wall ' time
Next State will be DrivingForwardToWall State
Mark that we are making a transition
Consume this event
Break of case RightRearBumpSensor

End of switch event type
End of if event is active statement
Break of RotatingCWToForward1

Case: If current state is DrivingForwardToWall
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has arrived to the wall
Start driving the robot backward
Reinitialize timer to 'backup to grab wall' time
Next State will be PushingBackUp State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

Case: RightFrontBumpSensor triggered event
Start driving the robot backward
Reinitialize timer to 'backup to grab wall' time
Next State will be PushingBackUp State
Mark that we are making a transition
Consume this event
Break of case RightFrontBumpSensor triggered

Case: LeftFrontBumpSensor triggered event
Start driving the robot forward
Reinitialize timer to ' backup to grab wall ' time
Next State will be PushingBackUp State
Mark that we are making a transition
Consume this event
Break of case LeftFrontBumpSensor
End of switch event type
End of if event is active statement
Break of DrivingForwardToWall

Case: If current state is PushingBackUp
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has done backing up
Start rotating robot clockwise

Reinitialize timer to '90 degrees rotation' time
Next State will be PushingRotation State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

End of switch event type
End of if event is active statement
Break of PushingBackUp

Case: If current state is PushingRotation
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has done rotating
Start driving the robot backward on a radius
Reinitialize timer for robot to 'grab the wall'
Next State will be CollectingRotatingWall State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

End of switch event type
End of if event is active statement
Break of PushingRotation

Case: If current state is CollectingRotatingWall
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has grabbed the wall
Start driving the robot backward on a radius
Reinitialize timer for ‘drive on radius’ time
Next State will be DrivingOnRad1 State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

End of switch event type

End of if event is active statement
Break of CollectingRotatingWall

Case: If current state is DrivingOnRad1
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has done driving on a radius without triggering any of the front bump sensors
Start driving the robot backward
Reinitialize timer to 'random motion backup' time
Next State will be RandomBallCollecting State
Mark that we are making a transition
Consume this event
End of if timeout statement
Break of case timeout

Case: RightFrontBumpSensor triggered event
Start driving the robot backward
Reinitialize timer to 'random motion backup' time
Next State will be RandomBallCollecting State
Mark that we are making a transition
Consume this event
Break of case RightFrontBumpSensor triggered

Case: RightUpperBumpSensor triggered event
Start driving the robot backward
Reinitialize timer to 'random motion backup' time
Next State will be RandomBallCollecting State
Mark that we are making a transition
Consume this event
Break of case RightUpperBumpSensor

End of switch event type
End of if event is active statement
Break of DrivingOnRad1

Case: If current state is RandomBallCollecting
If an event is active
Switch based on current event type
Case: timeout event
If it is ball collection timer, then robot has done backing up

Start rotating the robot
Reinitialize timer to 'random motion rotation' time
Consume this event
End of if timeout statement
Break of case timeout

Case: RightFrontBumpSensor triggered event
Start driving the robot backward
Reinitialize timer to 'random motion backup' time
Consume this event
Break of case RightFrontBumpSensor triggered

Case: LeftFrontBumpSensor triggered event
Start driving the robot backward
Reinitialize timer to 'random motion backup' time
Consume this event
Break of case LeftFrontBumpSensor

End of switch event type
End of if event is active statement
Break of DrivingOnRad1
End of switch states

If we are making a transition
Execute exit function for current state
Modify state variable
Execute entry function for new state
End if statement

Return ReturnEvent

End of RunBallCollectionSM

StartBallCollectionSM
Always start on RotatingToCenter state if enter without history
call the entry function (if any) for the ENTRY_STATE
return nothing
end of StartBallCollectionSM

QueryBallCollectionSM
Return Current State of BallCollectionSM
End of QueryBallCollectionSM

Sub-level State Machine 2: BallDumpSM

RunBallDumpSM

Assume no transition made
Assign NextState to CurrentState
Assume we are not consuming event

Based on current state, switch through the following cases

Case: if current state is FindingBinToDump
Query function OurAvailableBins() to know which bins are available
If an event is active
Switch based on current event type
Case: event is 20msBeaconDetected
If bin 1 is found to be an available bin
Start driving the robot forward towards the beacon
Set DefenseBin to be 1
Next State will be DrivingForwardToDump State
Mark that we are taking a transition
Consume this event
End if statement
Break of 20msBeaconDetected case

Case: event is 18msBeaconDetected
If bin 2 is found to be an available bin
Start driving the robot forward towards the beacon
Set DefenseBin to be 2
Next State will be DrivingForwardToDump State
Mark that we are taking a transition
Consume this event
End if statement
Break of 18msBeaconDetected case

Case: event is 16msBeaconDetected
If bin 3 is found to be an available bin
Start driving the robot forward towards the beacon
Set DefenseBin to be 3
Next State will be DrivingForwardToDump State
Mark that we are taking a transition
Consume this event
End if statement
Break of 16msBeaconDetected case

Case: event is 14msBeaconDetected
If bin 4 is found to be an available bin
Start driving the robot forward towards the beacon
Set DefenseBin to be 4
Next State will be DrivingForwardToDump State
Mark that we are taking a transition
Consume this event
End if statement
Break of 14msBeaconDetected case
End of switch event type
End of if event is active statement
Break of FindingBinToDump

Case: if current state is DrivingForwardToDump
If an event is active
Switch based on current event type
Case: If event is LeftUpperBumpSensor, then robot has hit the rotating wall, needs to look for available beacons again
Start rotating the robot clockwise
Next State will be FindingBinToDump State
Mark that we are taking a transition
Consume this event
Break of LeftUpperBumpSensor case

Case: If event is RightUpperBumpSensor, then robot has hit the rotating wall, needs to look for available beacons again
Start rotating the robot clockwise
Next State will be FindingBinToDump State
Mark that we are taking a transition
Consume this event
Break of RightUpperBumpSensor case

Case: if event is LeftFrontBumpSensor, then robot has hit the playing field wall
Initialize BALL_DUMP_TIMER to DUMP_BACKUP time
Start backing up the robot
Record the left bump sensor to be the last bump sensor triggered
Next State will be DumpBackUp State
Mark that we are taking a transition
Consume this event
Break of LeftFrontBumpSensor case

Case: if event is RightFrontBumpSensor, then robot has hit the playing

field wall
Initialize BALL_DUMP_TIMER to DUMP_BACKUP time
Start backing up the robot
Record the right bump sensor to be the last bump sensor triggered
Next State will be DumpBackUp State
Mark that we are taking a transition
Consume this event
Break of RightFrontBumpSensor case

End of switch event type
End of if event is active statement
Break of DrivingForwardToDump

Case: if current state is DumpBackUp
If an event is active
Switch based on current event type
Case: If event is timeout
If the left bump sensor is the last bump sensor triggered
Reinitialize BALL_DUMP_TIMER to be DUMP_ADJUST time
Start rotating the robot clockwise
Else if the right bump sensor is the last sensor triggered
Reinitialize BALL_DUMP_TIMER to be DUMP_ADJUST time
Start rotating the robot counter-clockwise

Initialize timer 5 to record DUMP_SWEEP time, last sweep along the outer radius before dumping
Next State will be DumpSweep State
Mark that we are taking a transition
Consume this event
Break of timeout case

End of switch event type
End of if event is active statement
Break of DumpBackUp

Case If current state is DumpSweep
If an event is active
Switch based on current event type
Case: LeftUpperBumpSensor triggered
If the left bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius

Else if the right bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Next State will be FindingTape State
Mark that we are taking a transition
Consume this event
Break of LeftUpperBumpSensor triggered

Case: RightUpperBumpSensor triggered
If the left bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Else if the right bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Next State will be FindingTape State
Mark that we are taking a transition
Consume this event
Break of RightUpperBumpSensor triggered

Case: FrontTwoBumpSensor triggered
If the left bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Else if the right bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Next State will be FindingTape State
Mark that we are taking a transition
Consume this event
Break of FrontTwoBumpSensor triggered

Case: if event is timeout event
Only respond to the BALL_DUMP_TIMER event
If the left bump sensor is the last bump sensor triggered
Start driving the robot forward along the outer radius
Else if the right bump sensor is the last bump sensor triggered
Start driving the robot forward along the outer radius
Only respond to timer 5 timeout event
If the left bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Else if the right bump sensor is the last bump sensor triggered
Start backing up the robot along the outer radius
Next State will be FindingTape State
Mark that we are taking a transition
Consume this event
Break of timeout event

End of switch event type
End of if event is active statement
Break of DumpSweep

Case: if current state is FindingTape
If an event is active
Switch based on the current event type
Case: if event is FrontTapeSensor detected
If the left bump sensor is the last bump sensor triggered
Start rotating the robot clockwise to align up with the tape
Else if the right bump sensor is the last bump sensor triggered
Start rotating the robot counter-clockwise to align up with tape
Next State will be AlignTape State
Mark that we are taking a transition
Consume this event
Break of FrontTapeSensor case
End of switch event type
End of if event is active statement
Break of FindingTape

Case: if current state is AlignTape
If an event is active
Switch based on the current event type
Case: if event is BackTapeSensor detected
Reinitialize BALL_DUMP_TIMER to record DUMP_BACKUP time
Start driving the robot backward
Next State will be ReverseDriving State
Mark that we are taking a transition
Consume this event
Break of BackTapeSensor case
End of switch event type
End of if event is active statement
Break of AlignTape

Case: if current state is ReverseDriving
If an event is active
Switch based on the current event type
Case: if event is RearTwoBumpSensor detected
Reinitialize timer to wait for dumping all the balls
Stop the robot for dump
Next State will be BallDumping State

Mark that we are taking a transition
Consume this event
Break of BackTapeSensor case

Case: if event is LeftRearBumpSensor detected
Start reverse driving only the Right wheel so that both rear bump sensors could be triggered
Consume this event
Break of LeftRearBumpSensor case

Case: if event is RightRearBumpSensor detected
Start reverse driving only the Left wheel so that both rear bump sensors could be triggered
Consume this event
Break of RightRearBumpSensor case

Case: if event is timeout
Only respond to the BALL_DUMP_TIMER timeout events
Stop the robot
Turn off fan
Reinitialize timer to wait for dumping all the balls
Next State will be BallDumping State
Mark that we are taking a transition
Consume this event
Break of Timeout case

End of switch event type
End of if event is active statement
Break of ReverseDriving

Case: if current state is BallDumping
If an event is active
Switch based on the current event type
Case: If event is Timeout
If event is BALL_DUMP_TIMER timeout
Start driving the robot forward for jerk motion
Reinitialize timer 5 for forward jerk motion time
Consume this event
Else if event is Timer 5 Timeout
Start driving the robot backward for jerk motion
Initialize timer 6 to record backward jerk motion time
Consume this event
Else if event is Timer 6 Timeout

Stop the robot
Wait for 2 seconds to go into the defense state
Mark that we are taking a transition
Post the event EV_FinishDumping
Break of timeout case

End of switch event type
End of if event is active statement
Break of BallDumping
End of switch states

If we are making a transition
Execute exit function for current state
Modify state variable
Execute entry function for new state
End if statement

Return ReturnEvent

End of RunBallDumpSM

StartBallDumpSM

Always start on FindingBinToDump state if enter without history
call the entry function (if any) for the ENTRY_STATE
return nothing
end of StartBallDumpSM

QueryBallCollectionSM

Return Current State of BallDumpSM
End of QueryBallDumpSM

Sub-level State Machine 3: BallDefenseSM

RunBallDumpSM

Assume no transition made
Assign NextState to CurrentState
Assume we are not consuming event

Based on current state, switch through the following cases

Case: if current state is InitialDefensePosition
If an event is active
Switch based on current event type
Case: event is WallApproachCW
Drive the robot forward and slightly CW
Initialize timer so that times out when drive to edge
Next State will be DrivingForwardCW State
Mark that we are taking a transition
Consume this event
End of case WallApproachCW

Case: event is WallApproachCCW
Drive the robot forward and slightly CCW
Initialize timer so that times out when drive to edge
Next State will be DrivingForwardCCW State
Mark that we are taking a transition
Consume this event
End of case WallApproachCCW
End of switch event type
End of if event is active statement
Break of InitialDefensePosition

Case: if current state is DrivingForwardCCW
If an event is active
Switch based on current event type
Case: event is timeout for BALL_DEFENSE_TIMER
Stop robot
Next State will be GuardLeftSide State
Mark that we are taking a transition
Consume this event
End of case timeout
End of switch event type
End of if event is active statement

Break of DrivingForwardCCW

Case: if current state is DrivingForwardCW
    If an event is active
        Switch based on current event type
            Case: event is timeout for BALL_DEFENSE_TIMER
                Stop robot
                Next State will be GuardingRightSide State
                Mark that we are taking a transition
                Consume this event
            End of case timeout
        End of switch event type
    End of if event is active statement
Break of DrivingForwardCW

Case: if current state is GuardingLeftSide
    If an event is active
        Switch based on current event type
            Case: event is WallLeaveCW
                Drive the robot reverse and slightly CW (i.e. Left wheel < Right wheel)
                Initialize timer so that times out when drive to initial defense position
                Next State will be DrivingReverseCW State
                Mark that we are taking a transition
                Consume this event
            End of case WallLeaveCW
        End of switch event type
    End of if event is active statement
Break of GuardingLeftSide

Case: if current state is GuardingRightSide
    If an event is active
        Switch based on current event type
            Case: event is WallLeaveCCW
                Drive the robot reverse and slightly CCW (i.e. Left wheel < Right wheel)
                Initialize timer so that times out when drive to initial defense position
                Next State will be DrivingReverseCCW State
                Mark that we are taking a transition

Consume this event
End of case WallLeaveCCW
End of switch event type
End of if event is active statement
Break of GuardingRightSide

Case: if current state is DrivingReverseCCW
If an event is active
Switch based on current event type
Case: event is timeout for BALL_DEFENSE_TIMER
Stop robot
Next State will be InitialDefensePosition State
Mark that we are taking a transition
Consume this event
End of case timeout

Case: event is RearTwoBumpSensor
Stop robot
Next State will be InitialDefensePosition State
Mark that we are taking a transition
Consume this event
End of case RearTwoBumpSensor

End of switch event type
End of if event is active statement
Break of DrivingReverseCCW

Case: if current state is DrivingReverseCW
If an event is active
Switch based on current event type
Case: event is timeout for BALL_DEFENSE_TIMER
Stop robot
Next State will be InitialDefensePosition State
Mark that we are taking a transition
Consume this event
End of case timeout

Case: event is RearTwoBumpSensor
Stop robot
Next State will be InitialDefensePosition State
Mark that we are taking a transition
Consume this event

End of case RearTwoBumpSensor

End of switch event type
End of if event is active statement
Break of DrivingReverseCW
End of switch states

If we are making a transition
Execute exit function for current state
Modify state variable
Execute entry function for new state
End if statement

Return ReturnEvents

End of RunBallDefenseSM

StartBallDefenseSM
Always start on InitialDefensePosition state if enter without history
call the entry function (if any) for the ENTRY_STATE
return nothing
end of StartBallDefenseSM

QueryBallDefenseSM
Return Current State of BallDefenseSM
End of QueryBallDefenseSM

State Machine: SPIFSM.c

InitSPIFSM

Save my priority
Put us into the InitSPIState
Post the initial transition event
If post event is true
Return true
Else return false
End of InitSPIFSM

PostSPIFSM

return ES_PostToService
end of PostSPIFSM

RunSPIFSM

Define the last SPI reading
Define the new SPI reading
Define the state reading
Define the new state reading
Assume no errors
Assign array number
Assign counter for command sending

Switch based on the current state
Case: If current state is InitSPIState
Only respond to EF_Init
Initialize FULL_READ_TIMER to SENDRECEIVE_TIME
Now put the machine into the SendRecieve State
Break of InitSPIState case

Case: If current State is SendReceive
Switch based on current event type
Case: if event is timeout
If timer is FULL_READ_TIMER
If transmit register is empty
Transmit data (query command) to master transmit register
Switch command based on the array number
Case 0: send AR_TOTAL_BALLS_ONE_SIDE
Case 1: send AR_BIN_1
Case 2: send AR_BIN_2
Case 3: send AR_BIN_3
Case 4: send AR_BIN_4

Case 5: send AR_WALL_ANGLE
End switch

If transmit data flag is set, read the junk value
Re-initialize BYTE_DELAY_TIMER to send out 0x00 to get real data
Else if timer is BYTE_DELAY_TIMER
If transmit register is empty
Send command 0x00
Re-initialize BYTE_DELAY_TIMER to read the real data received
Else if timer is RECEIVE_DELAY_TIMER
Read the real receive register

Assign NewReading to NewStateReading
If Array number is 0, AR_TOTAL_BALLS_ONE_SIDE
If current number of balls in the playing field is not 0
If last number of balls in the playing field is 0
Post event GameOn
Assign LastRading to NewReading
Assign StateReading to NewStateReading - 0x80
Else, Assign StateReading to NewStateReading – 1

Record StateReading to FieldStatus[] based on the current array number

Re-initialize FULL_READ_TIMER to send and recieve the next command
counter = counter +1

End switch on ThisEvent.EventType in SendReceive state
Break of SendRecieve State

End switch on Current State
Assign CurrentState to NextState

Return ReturnEvent
End of RunSPIFSM

InitSPI

Set baud rate divisor to 0x77, the slowest baud rate
Disable SPI interrupt

SPI system enable
Disable SPI transmit interrupt
Set SPI master/slave mode select as 1
Set SPI clock polarity to idel high
Set SPI clock phase to sample even edges
Slave select output enable
Set Most Significant Bit first
Set MODFEN, mode fault enable, high
End of InitSPI

Other Modules:

DrivePWM.c module

DrivePWM_Init

Set Port U0 and U1 as output ports
Enable PWM on U0 and U1 on E128
First clear the Prescale clk
Set Presale as /4
Set polarity of Port U0 and U1
Set to use the Scaled clock
Set Scale for 500Hz
No center align enabled
Set Period to 100
Enable PWM for Port U0 and U1
End of DrivePWM_Init

DrivePWM_SetDutyCycle

If the user defined motor is LEFT motor
Set Duty Cycle of PWMDTY0
Else if the user defined motor is LEFT motor
Set Duty Cycle of PWMDTY1
End of DrivePWM_SetDutyCycle

DriveRobot

Switch based on the user defined Movement
Case Forward:
Set Left Motor Dir pin low
Set Right Motor Dir pin high
Set Left Motor BRK pin low
Set Right Motor BRK pin low
Set Left Motor Duty cycle
Set Right Motor Duty cycle
Break of case forward

Case Backward:
Set Left Motor Dir pin high
Set Right Motor Dir pin low
Set Left Motor BRK pin low
Set Right Motor BRK pin low
Set Left Motor Duty cycle
Set Right Motor Duty cycle
Break of case backward

Case clockwise:
Set Left Motor Dir pin low
Set Right Motor Dir pin low
Set Left Motor BRK pin low
Set Right Motor BRK pin low
Set Left Motor Duty cycle
Set Right Motor Duty cycle
Break of case clockwise

Case counter clockwise:
Set Left Motor Dir pin high
Set Right Motor Dir pin high
Set Left Motor BRK pin low
Set Right Motor BRK pin low
Set Left Motor Duty cycle
Set Right Motor Duty cycle
Break of case counter clockwise

Case stop:
Set Left Motor Dir pin high
Set Right Motor Dir pin high
Set Left Motor BRK pin high
Set Right Motor BRK pin high
Set Left Motor Duty cycle
Set Right Motor Duty cycle
Break of case stop
End of DriveRobot

Beacon Sensing module

InitBeaconDetectionTimer
Set pin T0 as in input (Use TIM0_CLK4)
Enable the timer system
Set prescale to M/128
Sets PT0 to IC4 (set bit = 0)
Enable IC4 interrupt (PT0)
Capture rising edges
Clear IC4 flag
Enable global interrupts
End of InitBeaconDetectionTimer

BeaconInputCapture

Store period in local variable
Update last edge reading
Clear interrupt flag
Post events based on uPeriod
If uPeriod is within range of 20ms Beacon signal
Post 20ms beacon detected event
If uPeriod is within range of 18ms Beacon signal
Post 18ms beacon detected event
If uPeriod is within range of 16ms Beacon signal
Post 16ms beacon detected event
If uPeriod is within range of 14ms Beacon signal
Post 14ms beacon detected event

End of BeaconInputCapture